UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Electrònica Industrial i Automàtica

DESENVOLUPAMENT D'UN SISTEMA D'EXPLORACIÓ PER A ROBOTS MÒBILS

Memòria, Pressupost i Annexos

Autor: Jaume Alavedra Mas
Director: Sebastián Tornil Sin
Co-Director: Juan Andrade Cetto

Convocatòria: Juny 2018

Resum

En aquest treball de fi de grau es desenvolupa un sistema basat en visió per computador en el qual, a través una càmera adjunta a un robot mòbil que enfoca el sostre, el sistema és capaç de descriure els moviments i posició relativa al robot.

El desenvolupament d'aquest sistema respon a un projecte proposat per l'Institut de Robòtica i Informàtica Industrial en el qual es requereix una aplicació de baix cost econòmic capaç de localitzar, en una estança, la posició relativa d'una cadira de rodes. Per tal de dur a terme tal objectiu s'ha optat per un sistema d'adquisició d'imatges a través d'una càmera *webcam*. Aquest processa cada un dels fotogrames i n'extreu la posició relativa en un pla 2-D així com l'angle relatiu.

El sistema d'adquisició de les imatges així com el seu processament s'ha desenvolupat en l'entorn de MATLAB® i es proposen dues funcions. Una que processa i gràfica les imatges obtingudes de la *webcam* a temps real i una altra que realitza les mateixes accions però que obté les imatges d'un vídeo.

Finalment, es compara els resultats del sistema implementat en aquest treball amb el de dos altres sistemes d'odometria de precisions i preus superiors per a poder-ne fer una comparació i avaluació vàlides.

Resumen

En este trabajo de fin de grado se desarrolla un sistema basado en visión por computador en el qual, a través de una cámara adjunta a un robot móvil que enfoca el techo, el sistema es capaz de describir los movimientos y posición relativa al robot.

El desarrollo de este sistema responde a un proyecto propuesto por el Institut de Robòtica i Informàtica Industrial en el qual se requiere de una aplicación de bajo coste económico capaz de localizar, en una estancia, la posición relativa de una silla de ruedas. Para llevar a cabo tal objetivo se ha optado por un sistema de adquisición de imágenes a través de una cámara webcam. Este procesa cada uno de los fotogramas y extrae la posición relativa en un plano 2-D así como el ángulo relativo.

El sistema de adquisición de las imágenes, así como su procesamiento se ha desarrollado en el entorno de MATLAB® y se proponen dos funciones. Una que procesa y gráfica las imágenes obtenidas de la webcam en tiempo real y otra que realiza las mismas acciones pero que obtiene las imágenes de un vídeo.

Finalmente, se compara los resultados del sistema implementado en este trabajo con el de otros dos sistemas de odometría de precisiones y precios superiores para poder hacer una comparación y evaluación válidas.

Abstract

In this Bachelor Thesis a computer vision-based system is developed in which, through a camera attached to a mobile robot that focuses on the roof, the system is able to describe the movements and relative position of the device.

The development of this system responds to a project proposed by the Institut de Robotica I Informàtica Industrial, which required a low-cost application capable of locating the relative position of a wheelchair in a room. In order to carry out this objective, an image acquisition system that obtains frames through a webcam camera has been chosen. This processes each of the frames and extracts the relative position in a 2-D plane as well as the relative angle.

The image acquisition application, as well as its processing, has been developed in the MATLAB® environment, where two functions are proposed. One that processes and graphs the images obtained from the webcam in real time and another that performs the same actions but uses a video as input.

Finally, the implemented results of the system in this Thesis are compared with the ones from two systems with higher odometry precision in order to be able to make a valid comparison and evaluation of the developed solution.

Agraïments

A la meva família,
al director d'aquest treball,
a l'Institut de Robòtica i Informàtica Industrial
així com als seus tècnics i director.

Glossari

ROS	Robot Operating System
IRI	Institut de Robòtica i Informàtica Industrial
FOV	Field Of View
MAPE	Mean absolute percentage error
LED	Light Emitting Diode
VO	Visual Odometry
RGB	Red-Green-Blue

Llista de figures

Índex

1. Prefaci

En aquest capítol s'explicarà l'origen del treball, la motivació i requeriments previs.

1.1. Origen del treball

L'institut de Robòtica i Informàtica Industrial és un Centre de Recerca del Consell Espanyol per la Recerca Científica (CSIC) i la Universitat Politècnica de Catalunya (UPC). L'institut té tres objectius principals: promoure la recerca en el camp de la robòtica i la informàtica aplicada, cooperar amb la comunitat per al desenvolupament de projectes de tecnologia industrial i oferir educació científica a través de cursos.

Aquest institut du a terme diversos projectes, cadascun dels quals es conforma per diversos integrants. A raó d'una necessitat expressada per la Fundació Nexe es va decidir començar el projecte en el qual aquest treball de fi de grau estarà integrat. La Fundació Nexe és una institució benèfica amb seu a Barcelona dedicada a l'atenció organitzada a infants amb discapacitat greu. La cura d'aquests infants exigeix sacrificis per part dels pares així com de tota la família, tant emocionalment com econòmicament. L'alleujament que aquesta associació ofereix als nens també repercuteix emocionalment en els pares i altres familiars.

Les necessitats financeres de la Fundació Nexe han estat una preocupació contínua del seu director, el senyor Cécile de Visscher. No obstant això, independentment dels èxits de la Fundació Nexe, els costos de molts productes de cura tecnològica inevitablement recauen en els pares dels infants. Aquí és on es planteja part del repte.

L'objectiu d'aquest projecte és dissenyar un producte que pugui facilitar la vida a aquest grup específic de persones, de manera que els productes es puguin muntar en un Fab Lab (gairebé) per qualsevol. El repte tècnic és multidimensional: el programari serà de codi obert, mentre que les parts del producte físic (per exemple, sensors, actuadors, hidràulics, motors) han d'estar àmpliament disponibles al mercat o produïts de manera força fàcil amb l'equip digital i analògic de Fab Lab. A més a més, donat el focus particular en l'assemblatge no basat en una fàbrica, el projecte desafia les estratègies de disseny existents i el disseny per a les metodologies de muntatge.

1.2. Motivació

El repte de desenvolupar un sistema capaç d'emular funcionalitats d'altres que tenen un preu més elevat és, sens dubte, interessant. Igualment, que la solució adoptada pugui ser part de quelcom que faciliti la vida a aquells que el puguin necessitar dona una transcendència necessària a un projecte d'aquestes característiques.

A més a més, després d'haver cursat l'assignatura de Robòtica Industrial i Visió per Computador, vaig desenvolupar un gran interès pels sistemes basats en visió per computador així com les múltiples aplicacions que aquests tenen en el dia a dia.

1.3. Requeriments previs

Per tal de poder realitzar aquest treball de manera satisfactòria són necessaris, entre d'altres, els següents requeriments previs:

- **Coneixements de visió per computador.** És evident que, per tal de poder fer un tractament efectiu de cada un dels fotogrames és necessari saber quines són les operacions més adequades i optimes. A més a més, un coneixement de l'estat de l'art és de gran ajuda per tal de poder utilitzar, en cas necessari, les tècniques més actuals per a la resolució dels conflictes que puguin sorgir.
- **MATLAB®.** És l'eina emprada per a programar tot el sistema d'adquisició i processament dels fotogrames.

2. Introducció

En les últimes dècades, l'àrea de la robòtica mòbil i els sistemes autònoms han atret bastant l'atenció d'investigadors d'arreu del món, resultant en grans avenços. Actualment, els robots mòbils ja són capaços de realitzar tasques complexes de manera completament autònoma, mentre que en el passat la interacció d'un humà en molts casos era necessària. La robòtica mòbil té aplicacions en molts camps com per exemple el militar, el mèdic, l'aeroespacial, el de l'entreteniment o el domèstic. En aquests camps els robots han de realitzar tasques complexes que requereixen sistemes de navegació en espais on les condicions canvien de manera dinàmica tant siguin en llocs oberts o en tancats. En aquestes condicions els robots han de ser capaços de planificar la ruta que seguiran així com poder localitzar-se en l'entorn.

Per a poder realitzar aquesta última tasca, localitzar-se en l'entorn, existeixen una gran quantitat de tècniques que s'han proposat per tal de solucionar aquest atzucac. En aquest projecte es proposa una tècnica que aplica odometria visual per tal de solucionar aquest repte.

2.1. Objectius del treball

L'objectiu principal d'aquest projecte és el d'emular, amb un sistema basat en visió per computador, altres sistemes de descripció de traçat i localització més cars i menys accessibles per al consumidor comú. Per a dur a terme tal tasca, es discerneixen els següents objectius:

- **Desenvolupar un sistema de processament de fotogrames.** Amb la intenció de tractar cada un dels fotogrames obtinguts fins a l'últim pas d'aplicar la transformada de distàncies.
- **Desenvolupar un sistema capaç de descriure posició i angle relatius.** És el procés necessari descriure la posició relativa del robot en tot moment, es desenvolupa un sistema que descriu en un pla 2-D, la variació en centímetres de l'eix d'abscisses, l'eix d'ordenades i l'angle respecte la seva última posició.
- **Reproducció del traçat.** A través de les distàncies i angles relatius obtinguts es mostra manera gràfica els resultats del processament dels fotogrames es mostra la trajectòria del robot des del principi
- **Comparació del resultat.** Per tal de poder concloure si el sistema desenvolupat en aquest treball és fible pel que fa a la seva funció de localitzar el robot així com la seva traçada, es compara amb la trajectòria obtinguda amb dos sistemes d'odometria de característiques diferents.

2.2. Abast del treball

El desenvolupament d'un sistema d'exploració autònom per a robots mòbils és un tema que, a part de ser molt complex, engloba diversos camps cadascun dels quals dona en si mateix per un treball de fi de grau. És per això que aquest projecte se centra en el sistema de descripció del moviment i angle relatius així com en el traçat realitzat pel robot, també anomenat odometria. A més a més, hi ha una sèrie de consideracions que s'han definit per tal d'acotar un treball realitzable en el temps establert:

- **Acotar les condicions de funcionament.** En la fase de desenvolupament del codi necessari per a l'obtenció de les distàncies i angle relatius, s'estableix que per tal que el sistema funcioni correctament, aquest necessitarà funcionar en estances on hi hagi llums al sostre. Aquestes són necessàries per a poder determinar la distància recorreguda.
- **Establir les situacions que el sistema podrà descriure.** Per tal d'acotar una de les situacions més crítiques del sistema com ho són el tractament dels angles relatius, s'estableix que només es mostraran en la gràfica de la trajectòria aquells angles que siguin o bé 90 o bé 180 graus. Tanmateix, tots els angles són calculats i aquesta limitació només afecta en la traçada de la trajectòria en una gràfica.
- **Definir una configuració de càmera homogènia en tot el projecte.** Perquè el sistema de descripció de la traçada funcioni correctament s'estableix que serà necessari que l'objectiu de la càmera estigui col·locat de manera paral·lela al sostre.

3. Estat de l'art dels sistemes d'odometria

Com a estat de l'art s'entén quines són les pràctiques més modernes que es duen a terme en el cap d'estudi. A causa dels diversos camps que aborda el treball, aquest se centrarà en l'estudi de l'estimació de posició de vehicles amb rodes durant la navegació.

3.1. Què és l'odometria?

Per tal de poder estimar la seva posició en l'espai, els robots mòbils utilitzen sistemes d'estimació de la posició amb els que obtenen el desplaçament relatiu respecte la posició inicial. D'altra banda, també hi ha sistemes que obtenen el desplaçament relatiu a l'última posició coneguda del robot. La diferència entre els dos sistemes recau en el preu i en la precisió d'aquests.

La raó principal per a la qual els sistemes que detecten la posició relativa respecte a les últimes coordenades conegudes del robot són relativament econòmics d'implementar és que aquests, tot i tenir bona precisió a curt termini, a llarg termini acumulen errors que en desmereixen la precisió, fent-los així inviables per a bastants aplicacions que requereixen precisions mil·limètriques.

Mentre que inicialment la paraula odometria només es referia a l'estudi de la informació obtinguda de la rotació de les rodes per a poder estimar la posició en el temps, actualment és un terme que s'empra per descriure aquells sistemes que detecten la distància que ha recorregut un vehicle en un temps determinat no necessàriament basant-se en informació provinent de la rotació de les rodes.

3.2. Sistemes d'odometria

Actualment existeixen una gran quantitat de sistemes que empren l'odometria de maneres molt dispars. És per això que d'entre tots els sistemes d'odometria disponibles se'n dividirà l'estudi entre aquelles topologies que utilitzen el codificador de les rodes, les que utilitzen una seqüència d'imatges de l'entorn per estimar el moviment i per últim les que utilitzen sistemes basats en llum ultraviolada en entorns controlats.

3.2.1. Odometria mitjançant codificadors situats a les rodes

Aquesta és la més senzilla de les odometries. Utilitza equacions del moviment per tal de convertir les dades de les revolucions obtingudes en els codificadors de les rodes en desplaçament lineal relatiu a terra. Que el desplaçament sigui relatiu a l'última posició coneguda del robot indica que el sistema mai

podrà tindre una validesa absoluta en una varietat de casos. D'entre les causes d'imprecisions més comunes en aquest sistema se'n destaquen dos tipus: els errors sistemàtics i els errors no sistemàtics.

Els errors sistemàtics són aquells que quan ocorren es prolonguen durant tot el temps d'operació del robot mòbil. Són errors sistemàtics doncs un mal alineament de les rodes o la utilització de rodes de diàmetres dispars, entre d'altres.

D'altra banda, els errors no sistemàtics són aquells que només ocorren en certes ocasions. Exemples d'errors no sistemàtics són el patinatge de les rodes sobre el terra, obtenint així lectures de revolucions de la roda incorrectes, o el desplaçament sobre objectes inesperats que es trobin en el terra, entre d'altres.

És aquesta variabilitat en la certesa dels resultats que s'obtenen mitjançant aquest mètode el que obliga a la utilització de sistemes més fiables per a aquelles aplicacions que requereixen millors precisions.

3.2.2. Sistemes d'odometria visual

D'acord amb la definició que es dóna en la referència [3], els sistemes basats en odometria visual o VO (Visual Odometry) estimen el moviment de la càmera en temps real utilitzant imatges seqüencials com a entrada al sistema. Tal com es demostra amb l'aplicació que es planteja en la referència [2], a l'igual que els sistemes d'odometria basats en els codificadors de les rodes d'un robot mòbil, a través dels sistemes d'odometria visual també s'obtenen desplaçaments relatius a l'última posició coneguda del robot i, per tant, també són sistemes a través dels quals no és possible obtenir precisions mil·limètriques en proves molt llargues perquè acumulen error en cada mesura. Els sistemes d'odometria visual es divideixen en odometries visuals monoculars i odometries visuals estèreo.

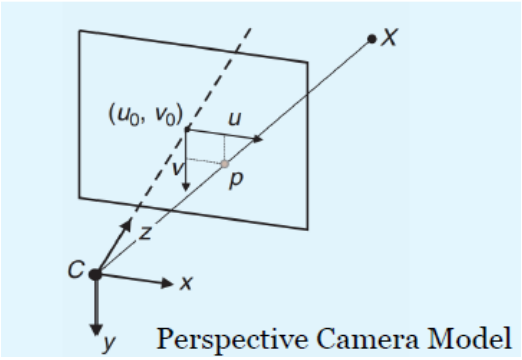

Figura 3.2. Sistema d'odometria visual monocular (font[4]).

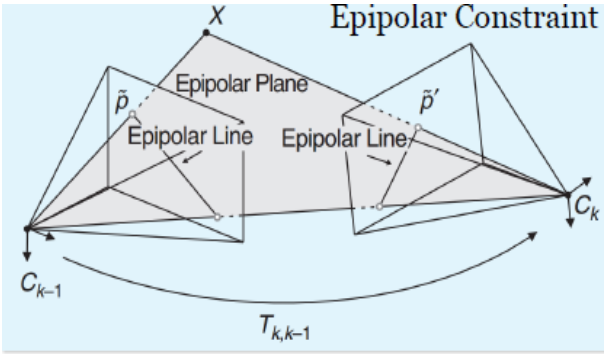

Figura 3.1. Sistema d'odometria visual *sterero* (font[4]).

En el sistema que es mostra en la Figura 3.2 s'hi observa com amb la tècnica d'odometria visual monocular no és possible obtenir l'escala ni la distància a l'objecte que apareix en el fotograma. És per això que aquest tipus de sistema s'acostuma a utilitzar de manera conjunta amb altres tècniques d'odometria.

D'altra banda, en el sistema de la Figura 3.1 sí que és possible obtenir l'escala i la distància l'objecte desitjat gràcies a la capacitat de discernir profunditats que proporciona l'ús de dues càmeres. Tanmateix, en alguns casos aquest sistema no es pot utilitzar i se'n simplifica la implementació utilitzant solament una de les dues càmeres disponibles.

Tal com es planteja en la referència [5], els sistemes que utilitzen la tècnica d'odometria mitjançant la visió segueixen el diagrama de flux de la Figura 3.3. En aquest hi conta una de les característiques més destacables d'aquests sistemes i d'on en deriven la quantitat més gran de problemàtiques en aplicacions en la realitat, el procés de seleccionar punts especials (*features)* o característiques del fotograma que es repeteixin en fotogrames consecutius (*feature matching)* per a poder calcular el desplaçament d'aquests.

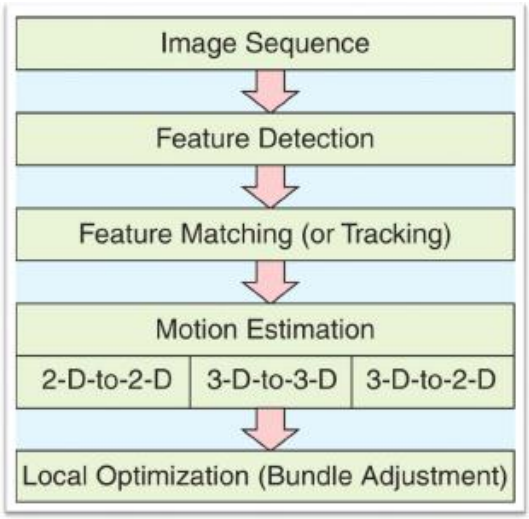

Figura 3.3. Diagrama de funcionament del sistema d'odometria visual (font[5]).

Tal com es discuteix en la referència [4], els problemes principals dels sistemes que utilitzen aquesta tècnica són, entre d'altres, els següents:

- La localització en entorns on la quantitat de característiques distintives és limitada.
- La localització en entorns on les condicions d'il·luminació no són de moderades a bones.
- Condicions en les quals el robot es mou a una velocitat massa elevada, fet que ocasiona distorsió en els objectes que apareixen en els fotogrames.

3.2.3. Sistemes d'odometria basats en llum ultraviolada

Les tècniques d'odometria basades en llum ultraviolada són de lluny les més precises del mercat actualment, sent capaces d'oferir precisions de menys de 20 µm en condicions òptimes. La raó principal per la qual aquests sistemes són capaços d'oferir tals precisions és que detecten la posició del robot relativa a la coordenada en la qual el robot ha començat la prova. Això fa que, l'error que pugui haver-hi en cada mesura no s'acumuli al llarg de tota la prova sinó que es manté constant per a cada nova mesura.

La problemàtica principal amb aquest tipus de sistema és que requereixen una instal·lació prèvia de les de càmeres així com aïllament de la llum solar quan vulgui utilitzar-se, ja que aquesta conté ones electromagnètiques que ocasionen falses lectures a les càmeres.

A més a més, el preu dels productes necessaris perquè aquesta tècnica funcioni és molt elevat i de difícil implementació. És per això que no es considera aplicable en el mercat del consumidor comú.

Aquests sistemes basen el seu funcionament en la instal·lació d'una sèrie de càmeres amb LEDs que emeten llum ultraviolada. Essen aquest tipus d'ona altament reflectant a certs materials, es col·loca un objecte (o sèrie d'objectes) amb aquestes característiques en l'àrea on s'han instal·lat les càmeres i els LEDs. Així doncs, a través de les càmeres s'obtenen els fotogrames on apareixen els objectes que reflecteixen l'ona d'interès i, a través de programes informàtics, se'n discerneixen la posició respecte a una coordenada d'origen prèviament preestablerta.

4. Obtenció i tractament dels fotogrames

4.1. Col·locació de la càmera

Per tal d'obtenir els fotogrames desitjats pel tractament dels mateixos la càmera ha de ser col·locada en una configuració determinada. Aquesta configuració és constant durant tot aquest projecte.

Tal com s'aprecia en la Figura 4.1 i la Figura 4.2, l'objectiu de la càmera que estigui capturant les imatges ha d'estar en tot moment col·locat paral·lelament al sostre on la càmera enfoca. Amb aquesta col·locació determinada es pretén evitar els efectes de distorsió que ocorrerien en cas que la càmera estigués disposada en qualsevol altra configuració. A més a més, d'aquesta manera és possible assegurar-se que les llums que puguin estar col·locades en el sostre afectin de manera homogènia a l'obtenció dels fotogrames al llarg de les proves.

En la Figura 4.1 s'observa la configuració utilitzada en el laboratori de l'Institut de Robòtica i Informàtica Industrial. Aquesta es conforma per un robot Pioneer 3 – AT en el qual se l'hi ha instal·lat una càmera Logitech E3560 QuickCamera amb una resolució de 640x480. Aquesta configuració serà la utilitzada per tal d'obtenir les odometries amb les que es compararà el sistema desenvolupat en aquest treball.

En la Figura 4.2 s'hi observa la configuració que s'utilitza per a provar els algorismes desenvolupats en ambients i condicions similars a les que podrien trobar-se les persones les quals utilitzarien aquest sistema. Aquesta es conforma per un robot iRobot Roomba 880 en el qual se l'hi ha instal·lat un mòbil LG Nexus 5, del qual s'obtindran fotogrames amb una resolució de 720x480.

S'aprecia doncs que aquest és un sistema d'odometria visual monocular al tractar-se ambdues configuracions d'una sola càmera.

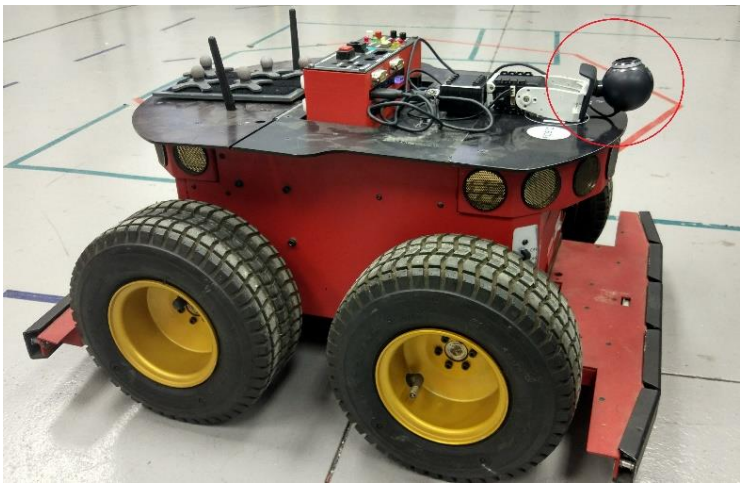

Figura 4.1. Fotografia de la posició de la càmera en el Robot Pioneer 3-AT.

Figura 4.2. Posició de la càmera en el robot iRobot Roomba 880.

4.2. Estudi dels mètodes necessaris en el tractament dels fotogrames

Per tal de poder discernir quins són els mètodes necessaris per a tractar de manera òptima cada un dels fotogrames, primerament es realitza l'anàlisi d'un sol fotograma. Amb aquesta anàlisi s'estableixen una sèrie de mètodes que s'apliquen, més endavant, a tots els fotogrames que l'algorisme processa.

4.2.1. Anàlisi de les fases de preprocessat i segmentació en un fotograma

Primerament obté una imatge digital basada en píxels. Aquest fotograma, com tota la resta, està basat en una composició de color en termes d'intensitat dels valors primaris de la llum, RGB. Cada píxel d'aquesta imatge serà convertit a escala de gris on, respecte a cada nivell de gris, se'n computarà la quantitat de píxels que hi són presents. Així doncs, cada un d'aquests píxels estarà codificat en un valor entre 0 i 255 (1 byte/píxel), sent el 0 un valor equivalent al negre i el 255 el blanc.

És aleshores quan ja és possible de traçar la gràfica que mostrarà la quantitat de píxels en cada nivell de gris, anomenada histograma. S'aprecia en la Figura 4.3 que en l'eix d'ordenades equival al nombre de píxels en un nivell de gris determinar i l'eix d'abscisses equival a cada un dels nivells de gris. Per tal de discernir si distribució de píxels cada nivell de gris és òptima es computa la corba d'energia.

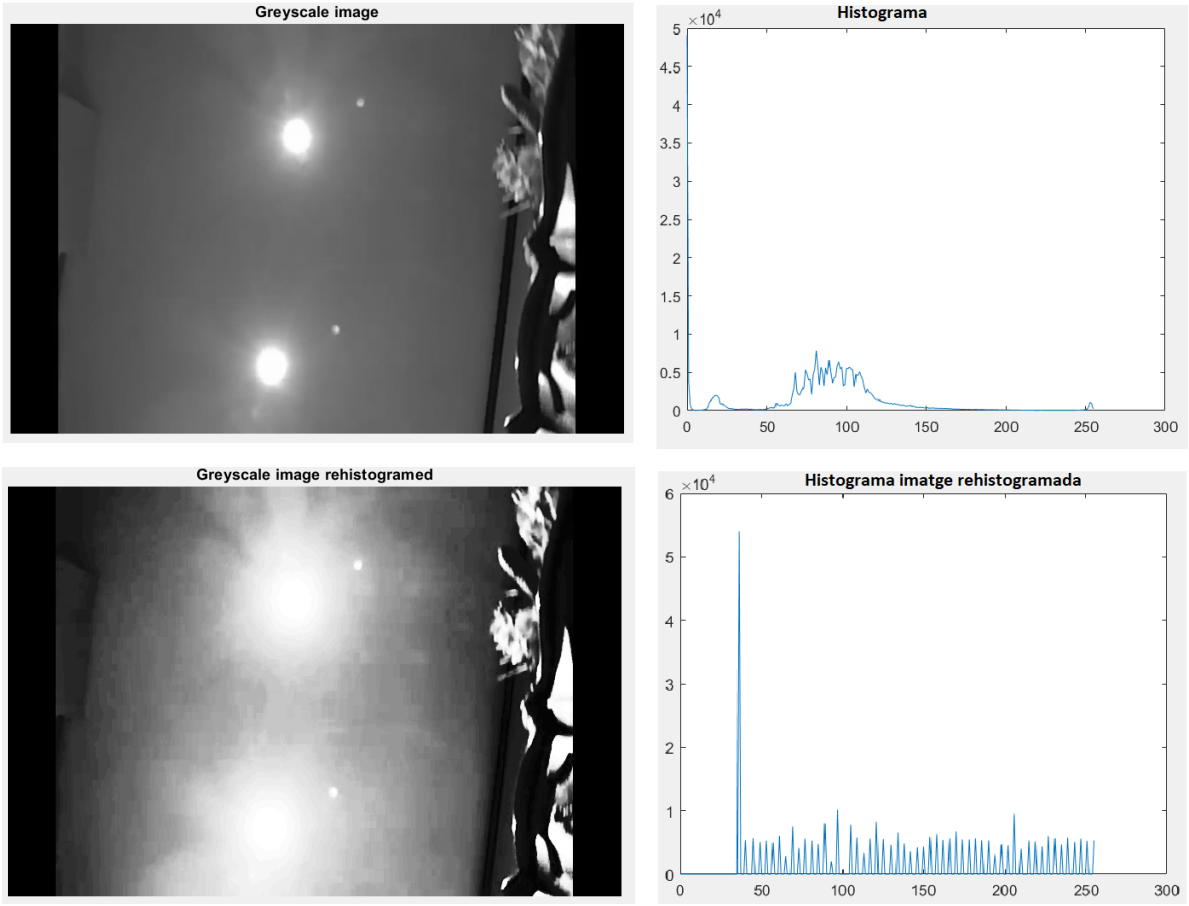

Figura 4.3. Comparació de la imatge original i la imatge rehistrogramada amb els seus respectius histogrames.

Tal com pot apreciar-se en la Figura 4.3, en l'histograma de la imatge original s'hi pot observar com el nombre de píxels en cada nivell de lluminositat no està equalitzat per tots els tons de gris. Això provoca que, tal com s'observa en la respectiva corba d'energia de la Figura 4.4, hi ha poca probabilitat d'aparició de píxels després d'aproximadament el nivell 150. Sabent que, en una imatge que estigui ben contrastada és necessari que la funció de transformació o corba d'energia tingui un pendent similar a la unitat, es pot afirmar que en l'histograma procedent de la imatge original transformada a escala de grisos hi ha marge de millora.

Per tal de millorar la imatge es realitza una rehistogramació del fotograma en escala de grisos. Tal com s'observa en la Figura 4.3, en el nou histograma s'aprecia com quasi tots els nivells de grisos presenten píxels, fet que provoca que la funció de transformació de l'histograma de la Figura 4.4 indiqui que hi ha probabilitat d'aparició en tots els nivells de gris a partir de quan la mateixa deixa de ser zero, sent així una imatge que està millor contrastada.

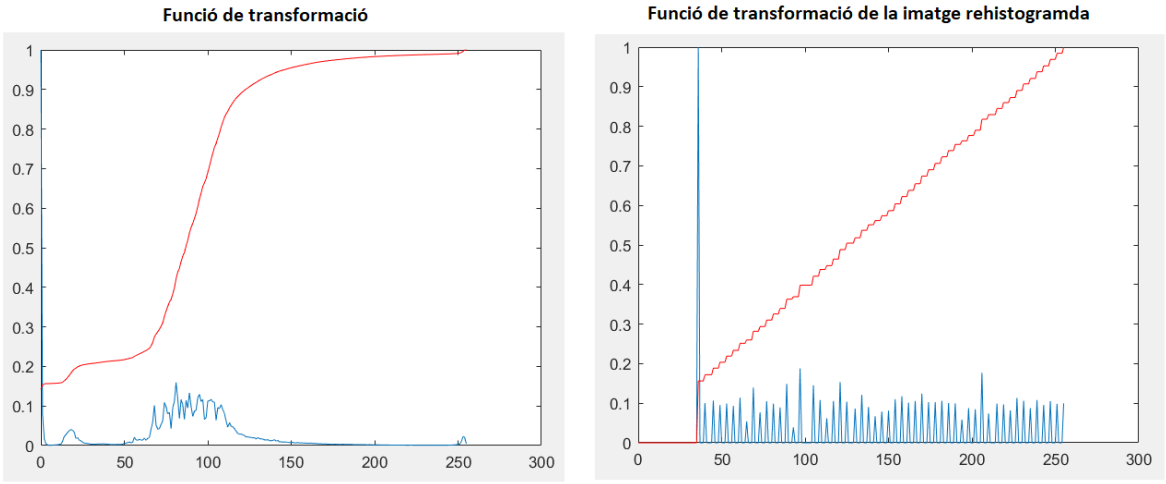

Figura 4.4. Funcions de transformació del fotograma a escala de grisos i de la imatge a escala de grisos rehistogramada.

```
%Lectura del vídeo
%------------------------------
ceilingVid = VideoReader('ceilingVideo.mp4');
get(ceilingVid)

%----------------------------
%PROCESSAT
%------------------------------
grayImage = rgb2gray(read(ceilingVid,361)); %conversió a greyscale a un ms determinat
figure(1), imshow(grayImage,{}),title('Greyscale image')
[FREQ,X]=imhist(grayImage); % crea l'histograma
FREQ=FREQ/max(FREQ);
figure(2),plot(X,FREQ)
acc(1)=FREQ(1);

for i=2:max(X)+1 %funció de transformació
acc(i)=acc(i-1)+FREQ(i);
end
hold on
plot(X,acc'/sum(FREQ),'r')
hold off

%-----Rehistogramació ----------------------
grayImage_eq=histeq(grayImage); %Equalitzatció de l'histograma
figure(3), imshow(grayImage_eq,{}),title('Greyscale image rehistogramed')
[FREQ,X]=imhist(grayImage_eq);
FREQ=FREQ/max(FREQ);
figure(4),plot(X,FREQ)
acc(1)=FREQ(1);

for i=2:max(X)+1 %funció de transformació
acc(i)=acc(i-1)+FREQ(i);
end
hold on
plot(X,acc'/sum(FREQ),'r')
hold off
```

Fragment de codi 4.1. Captura i pre-processament d'un fotograma.

Gràcies al Fragment de codi 4.1 s'obtenen les imatges presents en la Figura 4.3. Tot i això, els processos de rehistogramació no s'acostumen a utilitzar en sistemes automàtics, ja que en si mateixos, només

milloren l'aspecte visual de la imatge per a un observador humà i no suposen una millora substancial en el processament de les imatges. Tanmateix, són de gran ajuda per a poder apreciar els nivells de gris de la imatge així com el rang dinàmic d'aquesta.

Una de les operacions que sí que està basada en l'histograma i que és de gran utilitat en processament d'imatges molt contrastades és la binarització. Aquesta transformació és la principal raó per la qual s'ha aplicat aquest preprocessament a la imatge, ja que és una transformació que està destinada a reduir de la informació a solament dos nivells de gris, blanc i negre.

Tal com s'observa en el Fragment de codi 4.2, abans de passar a la fase de la segmentació, es decideix aplicar una operació morfològica d'obertura. Aquesta es conforma d'una operació d'erosió seguida d'una de dilatació mitjançant la qual s'aconsegueixen eliminar píxels aïllats i desconnectar aquelles regions que estiguin dèbilment unides.

```
%-----Obertura a 8 bits-------------------
s1 = strel('square',10);
grayImag_o8 = imopen(grayImage_eq,s1);
%-----Binarització------------------------
threshold = 0.95;
thresholdLevel = threshold * 255;
binaryImage = (grayImag_o8 >thresholdLevel);
figure(5), imshow(binaryImage,{}),title('Binarized image')
%----------------Erosió-----------------
s2 = strel('disk',20);
binaryImage_o = imopen(binaryImage, s2);
```

Fragment de codi 4.2. Última operació de pre-processament i operacions de segmentació.

Havent aplicat ja totes les operacions necessàries de la fase de preprocessament, comença la fase de segmentació. La segmentació és una fase clau en el reconeixement i/o localització d'objectes. El seu objectiu principal és la distinció de les regions presents en la imatge, on una regió és un conjunt de píxels connexos que presenten propietats semblants. Aquestes regions es corresponen amb objectes presents en la realitat capturada.

Tal com pot apreciar el lector, per tal de realitzar la tasca de binaritzar no s'ha emprat la funció pertinent ja integrada en MATLAB® anomenada "in2bw". La raó principal és que, en el supòsit que la tècnica desenvolupada en aquest treball resulti satisfactòria per al conjunt del projecte de l'IRI en la que aquest treball s'engloba, serà necessari que el codi desenvolupat en llenguatge de MATLAB® sigui convertible a o bé C o C++ per a poder-se integrar amb la resta de sistemes. Per tal de convertir aquest codi s'ha pensat d'utilitzar l'eina que ofereix MATLAB® anomenada MATLAB CODER. En la versió del programa emprada, MATLAB® 2018, aquesta eina té compatibilitat amb totes les funcions que s'utilitzen en el processament dels fotogrames excepte per a "in2bw". És per això que s'ha buscat una manera alternativa d'obtenir el mateix resultat.

En aquest projecte la fase de segmentació es basa en la binarització i l'etiquetatge. Primerament, tal com es mostra en el Fragment de codi 4.2, s'aplica la binarització. És conegut que les imatges es conformen per un rang de valor d'intensitat. A aquestes imatges se les anomena imatges de nivell de gris. Així doncs, la binarització, és el procés de conversió en el qual una imatge passa a estar conformada per només dos nivells d'intensitat (0 i 1). Després d'aquesta, les imatges passaran a ser representades per dos nivells de gris: el blanc i el negre.

Gràcies a la binarització dels fotogrames és possible utilitzar de manera òptima la memòria i la potència computacional del sistema que processa les imatges. A més a més, en aquestes imatges processades es mantenen les propietats geomètriques i topològiques dels objectes, fet que facilita l'obtenció d'aquestes. El procés de la binarització consisteix a comparar un valor llindar a cada píxel de la imatge que la conforma. Els píxels amb un valor igual o superior al llindar prenen el valor de "1", mentre que la resta prenen el valor de "0". En el Fragment de codi 4.2 s'hi observa com el llindar es defineix de manera manual en comptes d'obtenir-lo automàticament.

Del Fragment de codi 4.1 i del Fragment de codi 4.2 se'n deriva que l'operació de binarització s'aplica a la imatge en la qual s'ha equalitzat l'histograma. Quan s'obté el llindar automàtic de la imatge processada sense aplicar l'equalització de l'histograma el valor llindar és de 0,2039. Aquest valor tan baix (valor de gris 45,87) provoca el que s'observa en la Figura 4.5 i, en conseqüència amb l'histograma de la Figura 4.3, no és possible discernir els focus de llum del sostre.

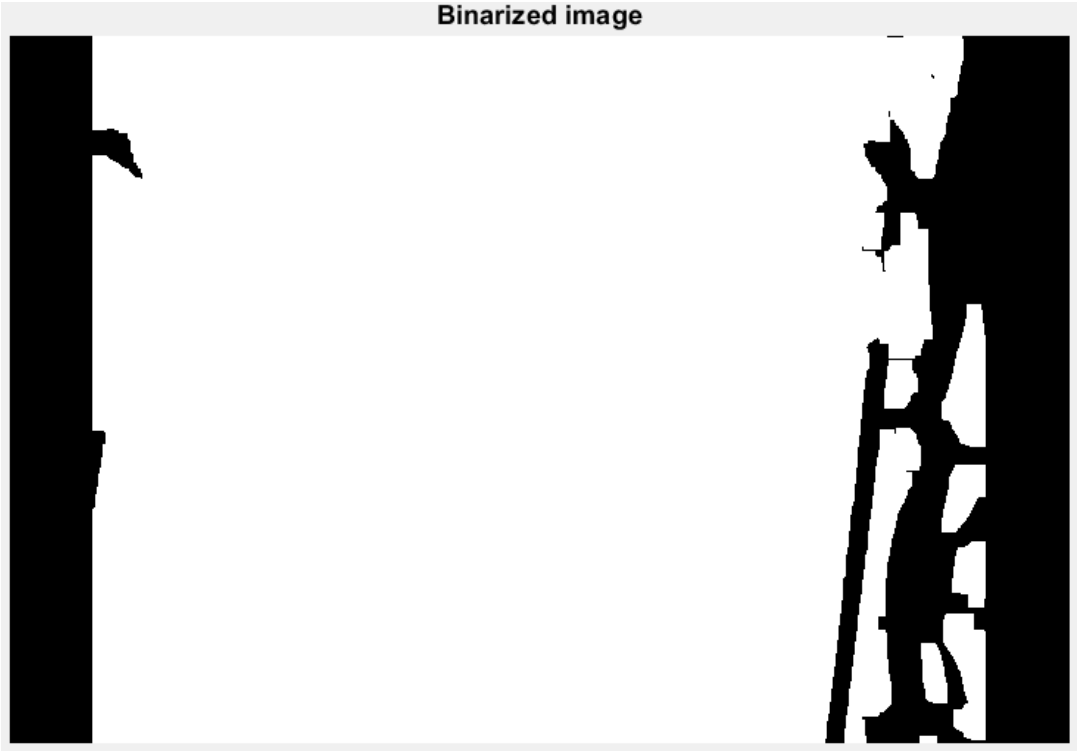

Figura 4.5. Imatge binaritzada amb llindar automàtic del fotograma.

Quan s'aplica un llindar automàtic sobre el fotograma sobre el qual s'ha equalitzat l'histograma, el llindar de binarització resulta ser d'un valor de 0.515. Aquest equival a 131,32 en valor d'escala de grisos, el qual tal com pot discernir-se en la Figura 4.6, no és suficient.

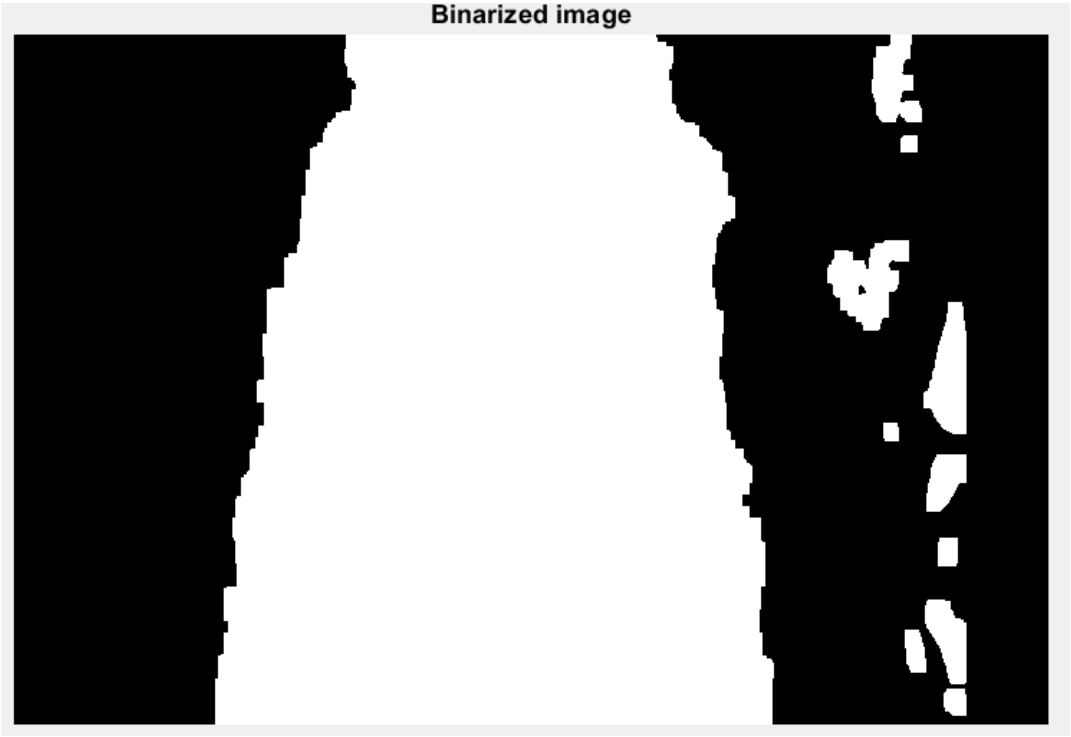

Figura 4.6. Imatge binaritzada amb llindar automàtic del fotograma on l'histograma s'ha equalitzat.

En aquest punt, i havent-se descartat la possibilitat d'aplicar un algorisme de binarització local per la gran potència computacional que seria necessària en cada fotograma, es decideix que s'aplicarà un valor llindar manual. Aquesta decisió tindrà un inconvenient que per l'abast d'aquest projecte s'ha considerat assumible, i és que en cas de canviar de condicions lumíniques el llindar s'haurà d'ajustar per obtenir un resultat òptim.

Analitzant doncs l'histograma de la imatge en la qual no s'ha equalitzat l'histograma de la Figura 4.3, s'observa com els píxels de major intensitat, els que estan més a prop del valor 255 de gris, estan clarament diferenciats de la resta i, per tant, és senzill decidir un valor llindar mitjançant el qual eliminar tots els píxels que no equivalen a focus de llum del sostre. L'inconvenient d'establir el valor llindar basant-se en aquest histograma és que s'està tenint en compte que els píxels equivalents al focus de llum del sostre sempre tindran un valor d'intensitat major clarament diferenciat de la resta. Aquesta condició s'ha pogut comprovar que no serà sempre certa degut, entre altres causes, al fet que en molts casos els focus de llum no enfoquen directament a la càmera i, per tant, la intensitat captada per la mateixa és destacablement menor. Aquests casos provoquen que l'histograma no formi una forma de campana tan diferenciada i que, aleshores, els píxels de major intensitat de l'histograma deixin de ser

propers al valor 255 de gris. Això deriva en què el llindar manual prèviament establert deixa de ser útil per no ser capaç de diferenciar les regions equivalents als focus de llum del sostre.

Per tal de supera aquest inconvenient el que s'ha comprovat ser més efectiu serà utilitzar la imatge en la qual s'ha equalitzat l'histograma. D'aquesta manera és possible assegurar-se que els píxels de màxima intensitat del fotograma sempre estaran més propers al valor de gris 255 i, per tant, podrà aplicar-se un valor llindar que discernirà els focus de llum en la gran majoria de les ocasions.

En el cas del fotograma que s'està tractant s'estableix un llindar del 95% dels valors de gris. De l'aplicació d'aquest llindar se'n deriva la Figura 4.7.

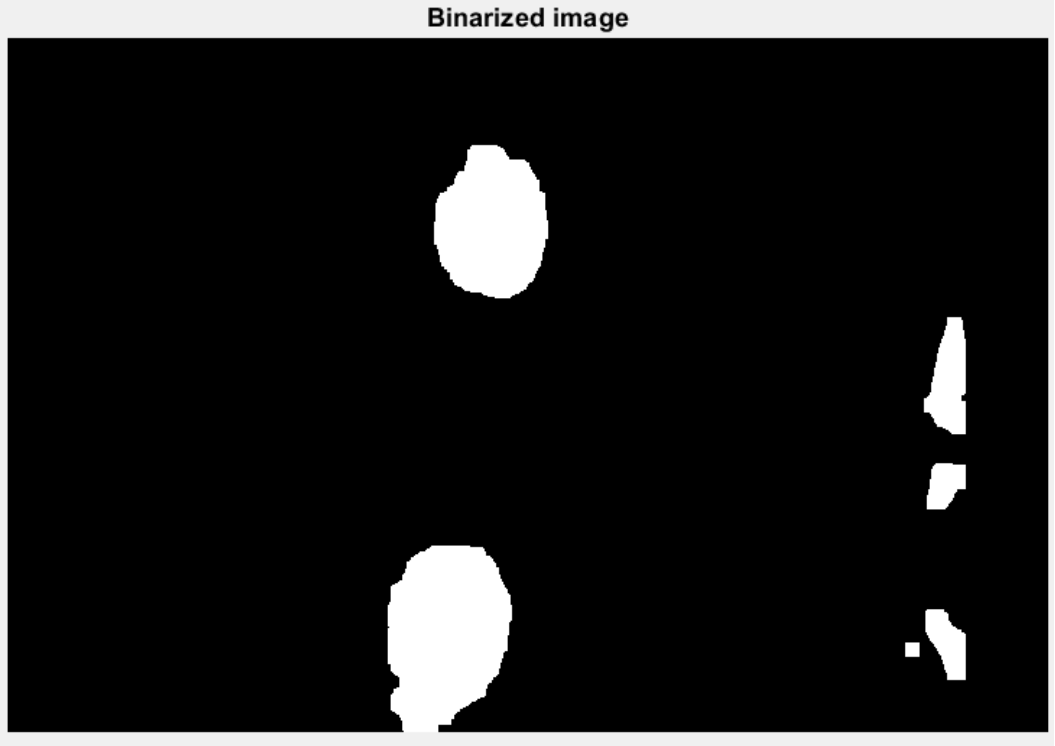

Figura 4.7. Imatge binaritzada amb llinar manual del fotograma on s'ha equalitzat l'histograma.

Havent aplicat la binarització s'aplica un procés d'erosió. L'erosió és una operació bàsica de morfològica que s'utilitza per separar regions dèbilment unides i per eliminar petits detalls. Havent aplicat l'erosió només queden les regions que estan significadament unides. S'aprecia en la Figura 4.8 com, havent aplicat aquesta operació, a les seccions que estan per sota del llindar establert se'ls hi suavitzen totes les vores d'acord amb les dimensions del disc amb el qual es practica el procés de l'erosió.

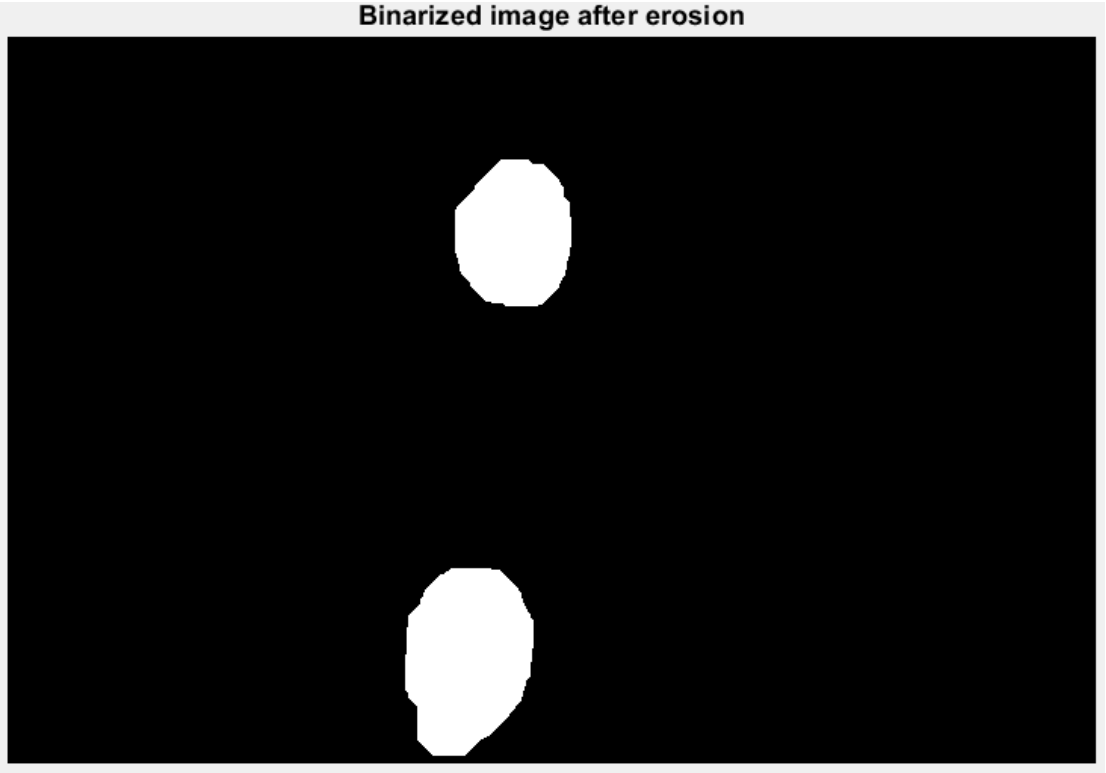

Figura 4.8. Imatge binaritzada després d'aplicar un procés d'erosió de disc amb radi 20.

Seguint les línies del Fragment de codi 4.2, l'últim pas de la fase de segmentació és l'etiquetatge. Aquesta operació identifica els conjunts connexos dels píxels a "1" i els hi associa un identificador únic per a cada conjunt. En la Figura 4.9 s'hi observen 3 regions diferents equivalents als que hi ha en la imatge binaritzada Figura 4.8. Una vegada aplicat el procés de l'etiquetatge s'assimila que les regions corresponen a objectes presents en la realitat capturada.

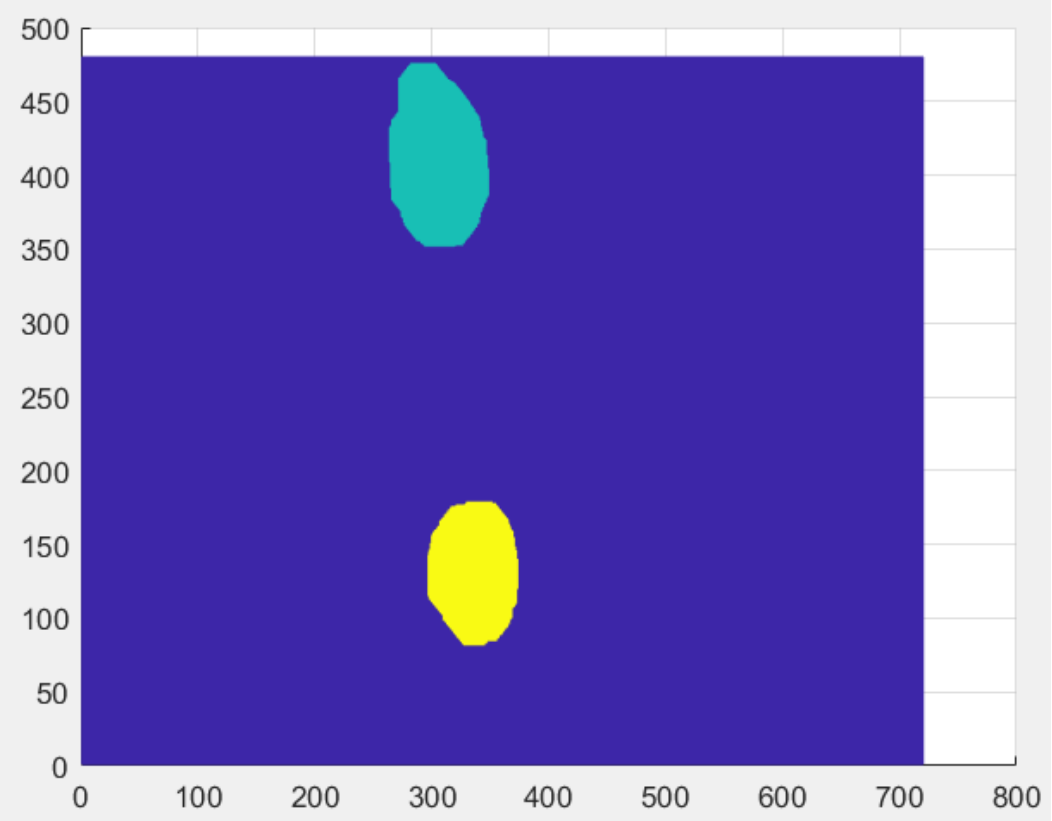

Figura 4.9. Gràfica de la imatge binaritzada etiquetada.

Per tal de determinar el moviment del robot respecte cada un dels objectes detectats s'aplicarà una tècnica anomenada transformada de distància. La transformada de distància és una representació derivada d'una imatge digital. El que aquesta fa és etiquetar cada un dels píxels amb un valor d'intensitat equivalent a la distància fins al píxel obstacle més proper. El píxel obstacle que s'utilitza en aquest projecte és el píxel que delimita les regions en el fotograma binaritzat.

Una vegada s'ha calculat la transformada de distàncies, es trobarà el píxel amb major intensitat de la regió amb l'àrea més gran i de la regió amb la segona àrea més gran. Tanmateix, no resultaria útil calcular la transformada de distàncies d'una imatge que té regions incompletes tocant els límits del fotograma, ja que com que no té tota la regió disponible per a buscar el píxel amb major intensitat, hi hauria la possibilitat que aquest encara hagués d'aparèixer i l'algoritme seleccionés un píxel incorrecte, fet que falsejaria la distància recorreguda pel robot.

Per tal d'evitar aquest problema, abans d'implementar la transformada de distàncies s'eliminaran totes les regions que estiguin tocant els límits del fotograma. El codi necessari per dur a terme tal acció pot observar-se en el Fragment de codi 4.3.

```
%-----------------Etiquetat---------------
[L,n]=bwlabel(binaryImage_o);
a = zeros(1,n);
for t=1:n
    for i=1:size(grayImage,1)
        for j=1:size(grayImage,2)

            if (L((j-1)*size(grayImage,1)+i))==t

                if
((i==1)||(j==1)||(i==size(grayImage,1))||(j==size(grayImage,2))&&(binaryImage_o(i,j)==1
))

                    a(t)=1;

                end
            end
        end
    end
end

%-------------------------------------------
%Eliminació de les regions que toquen les cantonades
for t=1:n
    for i=1:size(grayImage,1)
        for j=1:size(grayImage,2)

            if (L((j-1)*size(grayImage,1)+i))==t

                if (a(t)==1)

                    binaryImage_o(i,j)=0;

                end
            end
        end
    end
end
%-----------------Etiquetat------------------
[L,n]=bwlabel(binaryImage_o);
%----Distance transform----
binaryImage_DT = bwdist(~binaryImage_o);
```

Fragment de codi 4.3. Codi per l'eliminació de les regions que toquen les cantonades del fotograma.

Tal com s'observa en el Fragment de codi 4.3, és un programa que realitza un bucle iteratiu per cada una de les regions que s'han detectat prèviament amb el procés de l'etiquetatge. Així, comprovant si cada una d'aquestes regions té algun valor igual a les dimensions del fotograma, es poden guardar en la col·lecció de regions detectades que porta el nom de "a".

Havent detectat ja totes les regions que cal eliminar, es crea un altre bucle iteratiu que iguala la intensitat del píxel a zero en cas de detectar que estan dins de la col·lecció "a" prèviament creada.

Quan ja s'han eliminat totes les regions no desitjades és quan s'aplica el mètode de transforma de distàncies, tal com pot observar-se en la Figura 4.10.

Figura 4.10. Procés que mostra l'eliminació de les regions que toquen les cantonades i l'aplicació del mètode de la transformada de distàncies.

Una vegada ja es té el fotograma processat amb la transformada de distàncies es trobarà el punt dins de la regió desitjada que tingui la intensitat més gran. Aquest serà el punt que s'utilitzarà per a calcular la distància recorreguda pel robot respecte de la regió o regions detectades.

4.2.2. Aplicació de les fases de preprocessat i segmentació en vídeo i en temps real

Com s'esmenta en el Resum, en aquest projecte es presenten dos programes que realitzen les mateixes accions on la diferència entre els dos recau en com es realitza l'*input* de les imatges. Aquest pot fer-se a temps real, és a dir, el programa va donant la progressió del robot a mesura que aquest va movent-se, i l'altra és mitjançant vídeo, és a dir, gravar la progressió del robot i després processar les imatges simulant el sistema a temps real. En el transcurs d'aquest treball, les proves es realitzen mitjançant aquesta segona opció. La justificació principal per aquesta decisió és que, en cas que l'algorisme pogués necessitar algun tipus de millora en detectar alguna de les situacions, el programador pot modificar el codi i tornar a reproduir exactament les mateixes condicions que han ocasionat la necessitat de millora.

Tal com es mostra en el Fragment de codi 4.4, primerament es seleccionarà el vídeo a tractar. En cas que aquest no existeixi i MATLAB® no hagi pogut trobar la carpeta on aquest està ubicat, es mostrarà el missatge de la Figura 4.11. En aquest *pop-up* es dóna l'opció de o bé seleccionar un nou vídeo o cancel·lar l'operació.

```
% Selecciona la carpeta on hi ha el vídeo
folder = fileparts(which('Nom del vídeo'));
movieFullFileName = fullfile(folder, 'Nom del vídeo');
% Comprovació de si aquest existeix
if ~exist(movieFullFileName, 'file')
    strErrorMessage = sprintf('File not found:\n%s\nYou can choose a new one, or
cancel', movieFullFileName);
    response = questdlg(strErrorMessage, 'File not found', 'OK - choose a new
movie.', 'Cancel', 'OK - choose a new movie.');
    if strcmpi(response, 'OK - choose a new movie.')
        [baseFileName, folderName, FilterIndex] = uigetfile('*.avi');
        if ~isequal(baseFileName, 0)
            movieFullFileName = fullfile(folderName, baseFileName);
        else
            return;
        end
    else
        return;
    end
end
```

Fragment de codi 4.4. Selecció del vídeo a tractar i missatge d'error en cas de que aquest no existeixi.

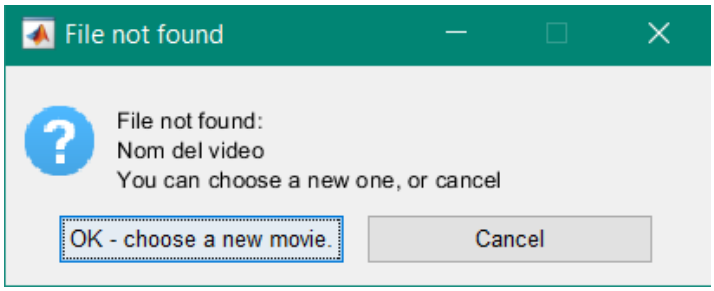

Figura 4.11. Error en localitzar el vídeo a processar.

El codi que s'empra en l'aplicació que processa els fotogrames d'un vídeo pel que fa al preprocessament i a la segmentació no canvia. El que sí que ho fa és la manera que té el programa d'anar seleccionant els fotogrames del vídeo que s'ha decidit processar. Per tal de dur a terme aquesta acció es crea un bucle que itera pels fotogrames, creat tal com es mostra en el Fragment de codi 4.5.

```
videoObject = VideoReader(movieFullFileName);
% Determine how many frames there are.
numberOfFrames = videoObject.NumberOfFrames;
vidHeight = videoObject.Height;
vidWidth = videoObject.Width;

for frame = 1:5: numberOfFrames
        thisFrame = read(videoObject, frame);
        % Display it
        hImage = subplot(2, 2, 1);
        image(thisFrame);
        caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
        title(caption, 'FontSize', fontSize);
        drawnow; % Force it to refresh the window.
        (...)
end
```

Fragment de codi 4.5. Bucle que itera en els fotogrames del vídeo.

S'observa en el Fragment de codi 4.5 que el bucle *for* que s'ha creat itera per tots els fotogrames del vídeo seleccionat en sumes de 5. Podria aplicar-se el mateix algorisme iteratiu tenint en compte tots els fotogrames del vídeo i processar-los un per un però, a part de ser un procés que allargaria en gran mesura el temps computacional, no resulta en una millora de resolució del sistema que en justifiqui el temps de processament. D'altra banda, en aquest fragment també s'hi observa com s'afegeix una visualització de les imatges a processar en el *workspace*.

```
cam = webcam('Logitech Webcam 300')

while(1)
    counter = counter + 1;
    ceiling = snapshot(cam);
    (...)
end
```

Fragment de codi 4.6. Selecció dels fotogrames en l'aplicació a temps real.

En el cas d'utilitzar l'algorisme dissenyat per treballar en temps real, els canvis necessaris queden plasmats en el Fragment de codi 4.6. En aquest, primerament se selecciona la càmera des d'on s'obtindran els fotogrames, seguidament es crea un bucle infinit en el qual s'incrementa el comptador de fotogrames processats i es captura una instantània de la imatge de vídeo que equival a un fotograma. A continuació el codi serà el mateix que el necessari per a processar les imatges obtingudes d'un vídeo. És per això que, en el transcurs d'aquest treball, l'aplicació que funciona a temps real no tornarà a esmentar-se entenent-se que aquesta funciona igual que la que obté els fotogrames d'un vídeo.

En el diagrama de blocs de Figura 4.12 s'hi aprecien les dues maneres implementades per tal d'introduir fotogrames en l'aplicació. Les dues són, pel que fa a la implementació real del codi, excloents perquè estan programades en aplicacions separades.

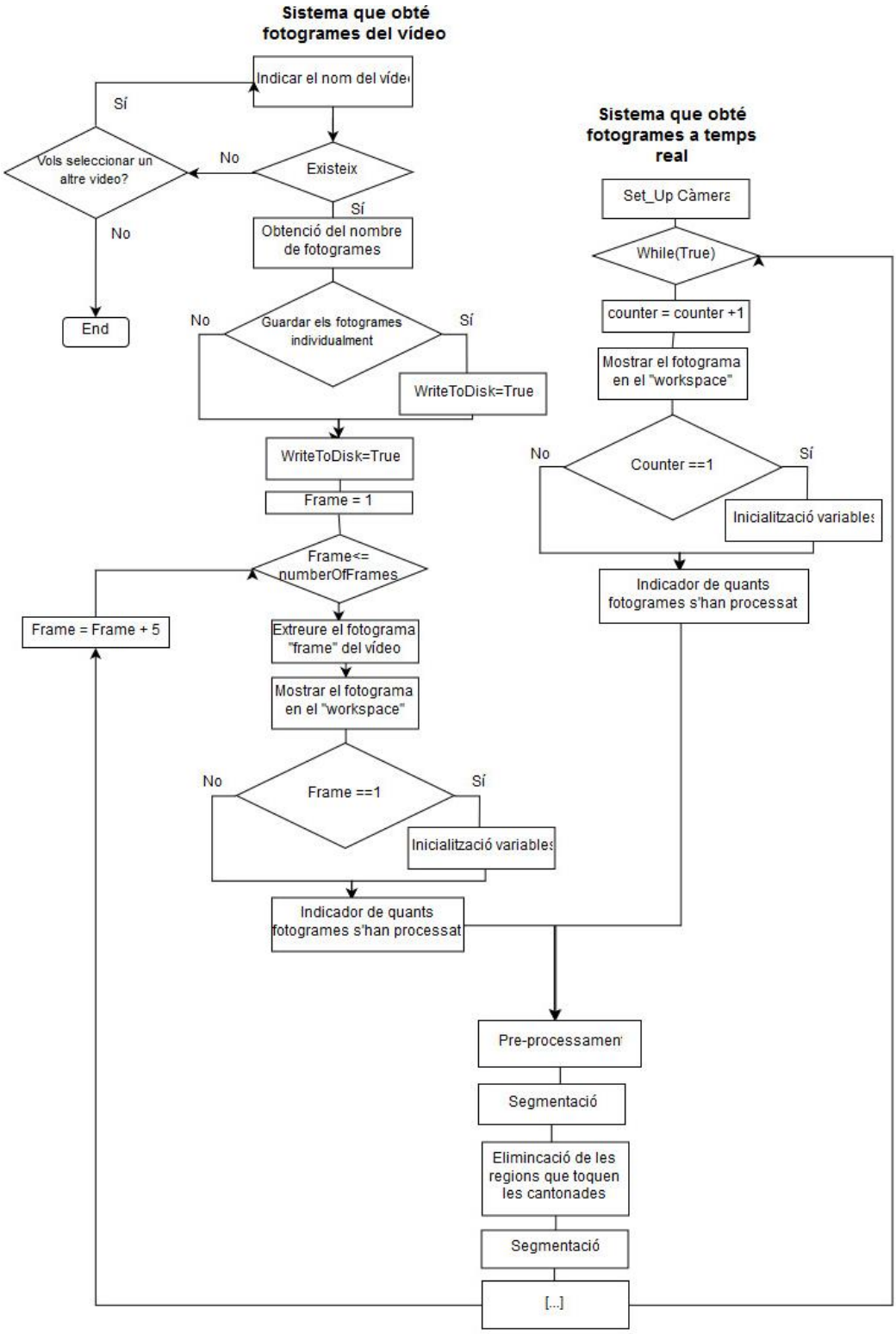

Figura 4.12. Diagrama de blocs del funcionament de l'aplicació abans d'entrar a la fase de preprocessat.

4.2.3. Bucle iteratiu per trobar el píxel de màxima intensitat

Quan ja s'han aplicat les fases de preprocessament i segmentació així com eliminat les regions no desitjades i aplicat la transformada de distàncies, es crea un bucle iteratiu que busca el píxel amb la intensitat més gran de les dues regions amb una àrea major presents en el fotograma.

```
% Display the distance transform image
subplot(2, 2, 3);
imshow(binaryImage_DT, {}),title('Distance Transform Image');
props = regionprops(binaryImage_o, 'area');
hold on;
area_gran = 0;
area_petita = 0;
label_gran = 0;
label_petita = 0;
if ((length(props) >= 1)&&(cont_first_blob == 0))
        first_blob = 1;
end
for k = 1 : length(props)
        area = props(k).Area;
        if area > area_gran
                area_petita = area_gran;
                label_petita = label_gran;
                area_gran = area;
                label_gran = k;
        elseif ((area > area_petita)&&(area < area_gran))
                area_petita = area;
                label_petita = k;
        end
        if (length(props) == 1)
                area_petita = 0;
        end
end
labels = [label_gran, label_petita];
positions_imatge = [pos_x_image_gran, pos_y_image_gran, pos_x_image_petita,
        pos_y_image_petita];
positions_imatge_update = [pos_x_image_gran_update, pos_y_image_gran_update,
        pos_x_image_petita_update, pos_y_image_petita_update];
intensities = [inte_gran, inte_petita];
pos_x_image_res = [pos_x_image_res_gran,pos_y_image_res_gran, pos_x_image_res_petita,
pos_y_image_res_petita];
for t=1:length(labels)
        for i=1:size(thisFrame,1)
                for j=1:size(thisFrame,2)
                     if (L((j-1) * size(thisFrame,1) + i)) == labels(t)
                         current_inte = binaryImage_DT(i,j);

                       if (intensities(t) < current_inte)
                            intensities(t) = current_inte;
                            positions_imatge(2 * t - 1) = j;
                            positions_imatge(2 * t) = i;
                       end
                end
        end
end
```

Fragment de codi 4.7. Bucle iteratiu per a trobar els píxels amb una intensitat més elevada.

Analitzant el Fragment de codi 4.7, primerament s'hi troba el codi responsable per a mostrar la imatge on s'hi ha aplicat la transformada de distàncies en el mateix *workspace* on també s'hi mostren la imatge original i la imatge binaritzada.

A continuació, s'obté l'àrea de les regions presents en el fotograma. Una vegada s'ha obtingut totes les àrees desitjades es crea un bucle que itera pel nombre total de regions i en troba les dues àrees majors. Havent trobat les àrees desitjades s'etiqueten les mateixes i es guarden dins d'una col·lecció que porta per nom *labels*. Seguidament es crea un altre bucle que itera per les dues etiquetes creades de l'àrea. Així, aquest últim bucle busca dins de la regió amb la primera i segona àrea més gran quin és el píxel que té la intensitat major i en guarda la posició en coordenades de cadascun d'ells.

5. Càlcul del desplaçament

Per a calcular el desplaçament real del robot respecte a les regions detectades és necessari primer calcular quin és el desplaçament respecte a la imatge del robot. Aquest donarà el número en píxels de la distància recorreguda entre dos fotogrames.

Tanmateix, el càlcul del desplaçament en la imatge a la pràctica no és tan senzill. És necessari tindre en compte una sèrie de situacions.

5.1. Situacions en què apareixen i desapareixen regions

1. Casos en els quals una regió desapareix d'un fotograma a un altre:

Analitzant la primera de les condicions *If* del Fragment de codi 5.1, s'hi observa que es detectarà que una regió a desaparegut d'un fotograma a una altre quan almenys un de les dues coordenades que descriu la posició del píxel de major intensitat de la regió en qüestió és diferent de zero en el primer dels fotogrames i en el segon els dos paràmetres que defineixen la posició passen a ser zero.

D'altra banda, comentar també que, tal com el lector podrà apreciar en el Fragment de codi 5.1, aquest comença amb una condició *If* que comprova que els càlculs següents no es realitzaran sobre el primer dels fotogrames capturats. La raó per prendre aquesta mesura és per donar temps a la càmera a adaptar-se a possible canvis de llum i enfocaments. Aquesta condició serà present des del fragment de codi esmentat fins que es comenti el contrari cap al final del programa.

S'observen dues possibilitats a tindre en compte pel que fa a aquest cas:

1.1- Cas en què la regió que desapareix correspon a l'àrea major

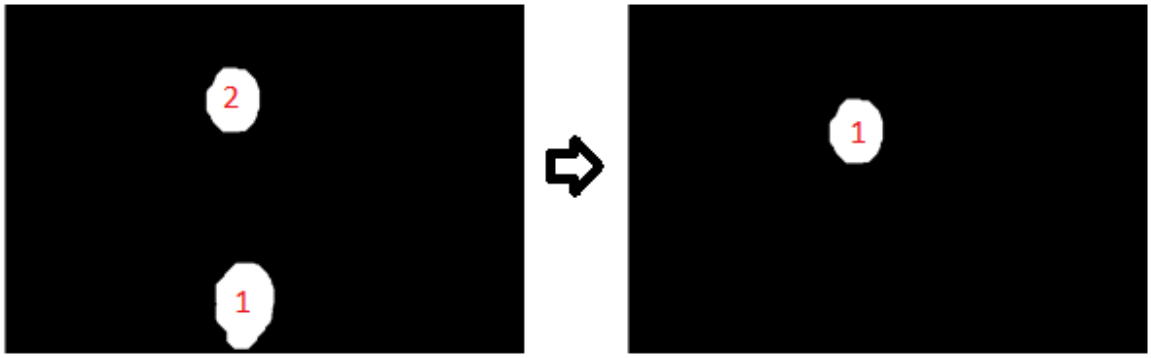

Figura 5.1. Cas de desaparició de la regió d'àrea major.

Tal com s'aprecia en la Figura 5.1, l'algorisme presentat en el Fragment de codi 4.7 etiqueta amb el número 1 la regió d'àrea major i amb el 2 la regió d'àrea menor. Així doncs, serà comú que la regió etiquetada amb un 1, o d'àrea major, desaparegui.

En aquest cas, la regió que tenia una àrea menor en el primer dels fotogrames passarà a ser la regió amb l'àrea major en el segon dels fotogrames en ser l'única regió present. Així doncs, serà necessari que les coordenades del píxel de major intensitat de la regió amb l'àrea menor del primer fotograma siguin les que s'utilitzin per comparar el desplaçament amb les coordenades del píxel de major intensitat de la regió del segon fotograma. Si no es dugués a terme cap acció, en aquest cas l'algoritme estaria comparant la distància recorreguda entre dos regions que no equivalen al mateix objecte entre dues fotogrames diferents i s'obtindria un resultat incorrecte.

Per tal de dur a terme les accions descrites es programa el que s'observa en el Fragment de codi 5.2. En aquest cas, compreses en la condició que mostra "Blob_Disappearence_Case_2" en la finestra de comandes quan aquesta s'executa.

1.2- Cas en què la regió que desapareix correspon a l'àrea menor

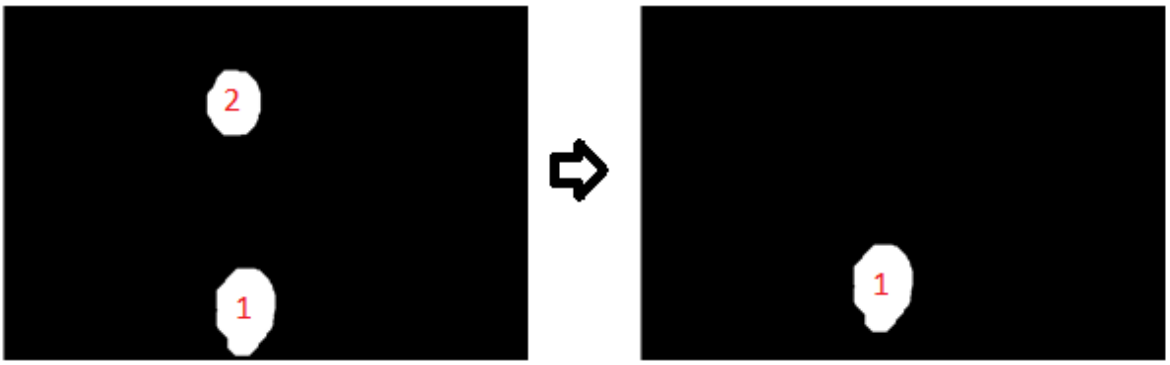

Figura 5.2. Cas de desaparició de la regió d'àrea menor.

En el cas que sigui la regió d'àrea menor la que desapareix, tal com es mostra en la Figura 5.2, les accions a dur a terme són més simples que en el cas anterior. La raó per això es que en aquest cas no és necessari canviar les coordenades de la posició del píxel de major intensitat de la regió amb una àrea major, ja que en els dos casos aquesta correspon al mateix objecte.

Això queda plasmat en la programació d'aquesta possibilitat que, tal com es mostra en el Fragment de codi 5.1, no es realitza cap intercanvi de coordenades. Aquesta condició pot ser identificada en el codi per ser el cas que mostra en la finestra de comandes la condició "Blob_Disappearence_Case_1".

Analitzant la condició *If* que detecta aquesta situació s'hi observa que en aquest cas es decideix utilitzar una de les propietats ja obtingudes prèviament, les àrees. Per tal de discernir quan es produeixen els dos possibles casos que s'han explicat s'empra una estructura *If/else* en la que es comprova que les dimensions de la regió d'àrea major del fotograma siguin similars a les dimensions de la regió de l'àrea major del fotograma immediatament anterior. Si aquestes són semblants, entenent que per semblants s'estableix que hi haurà com a màxim una variació del 20% en l'àrea total, s'estableix que la regió identificada amb l'àrea major no ha desaparegut.

```
if frame > 2
        if (((positions_imatge_update(2 * t - 1) ~= 0)||(positions_imatge_update(2 * t)
        ~= 0))&&(positions_imatge(2 * t - 1)==0)&&(positions_imatge(2 * t) == 0))

                if ((((area_update-(area_gran * 0.2)) <= area_gran)&&(area_gran <=
                area_update + (area_gran * 0.2)))&&(disapear == 0))
                        disp('--------------->>Blob-Disappearence_Case_1<<-----------')
                        one_blob=1;
                else
                        positions_imatge_update(3) = positions_imatge_update(1);
                        positions_imatge_update(4) = positions_imatge_update(2);
                        positions_imatge_update(1) = pos_x_image_petita_update;
                        positions_imatge_update(2) = pos_y_image_petita_update;
                        disp('--------------->>Blob-Disappearence_Case_2<<------------')
                        one_blob = 1;
                        apear = 1;
                        cont_apear = 0;
                end
        else
                cont_disapear= cont_disapear + 1;
                if (cont_disapear >= 4)
                        disapear = 0;
                        cont_disapear = 0;
                end
        end
```

Fragment de codi 5.1. Detecció i actuació en l'aparició de regions.

2. Casos en els quals una regió apareix d'un fotograma a un altre:

En aquest cas, per tal d'identificar quan una regió no identificada prèviament ha aparegut en un nou fotograma s'utilitza el contrari de les condicions emprades anteriorment en el cas de la desaparició d'una regió.

Com pot observar-se en el Fragment de codi 5.2, en la primera de les condicions *If* es comprova que, almenys, una de les dues coordenades que descriu la posició del píxel de major intensitat en una regió del fotograma que s'està processant sigui diferent de zero. Si aquesta condició es compleix i es confirma que les coordenades del fotograma anterior eren igual a zero aleshores pot afirmar-se que ha aparegut una regió nova prèviament no identificada.

S'observen dues possibilitats a tindre en compte pel que fa a aquest cas:

2.1- Cas en què la regió que apareix correspon a l'àrea major

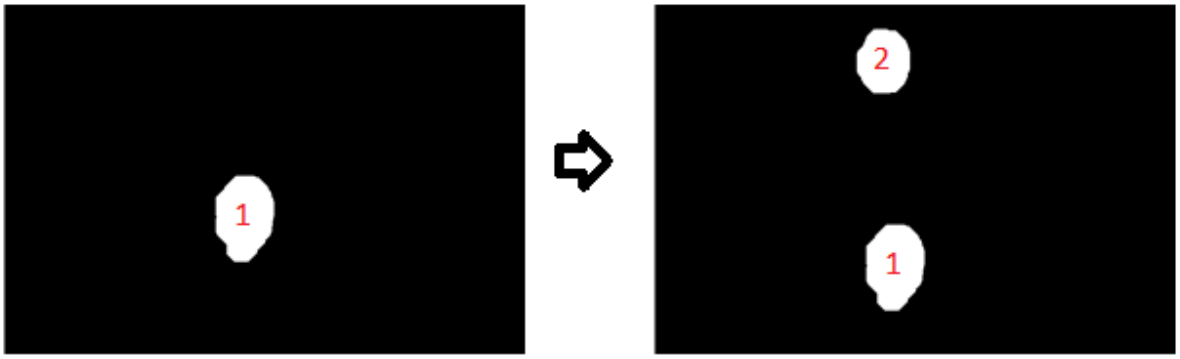

Figura 5.3. Cas d'aparició de la regió d'àrea menor.

Per tal de detectar aquest cas d'aparició d'una regió nova amb una àrea inferior a la regió ja coneguda, s'utilitza una tècnica semblant a l'emprada per tal de detectar els casos de desaparició de regions. A l'igual que en aquell cas, també s'utilitza com a marge màxim de diferència entre regions un 20% d'error. Tanmateix, en aquest cas aquest percentatge d'error que s'està disposat a acceptar s'aplica en l'àrea de la regió ja coneguda i no en l'àrea de la regió que ha aparegut, tal com es fa en el cas de la regió que desapareix.

D'altra banda, tal com s'observa en el Fragment de codi 5.2, en aquest cas no es realitza cap intercanvi de coordenades entre regions, ja que les coordenades de la regió amb l'àrea major equivalen a la mateixa regió del nou fotograma. Pot identificar-se ràpidament per ser l'opció que mostres "Blob_Appearence_Case_1" en la finestra de comandes.

2.2- Cas en què la regió que apareix correspon a l'àrea menor

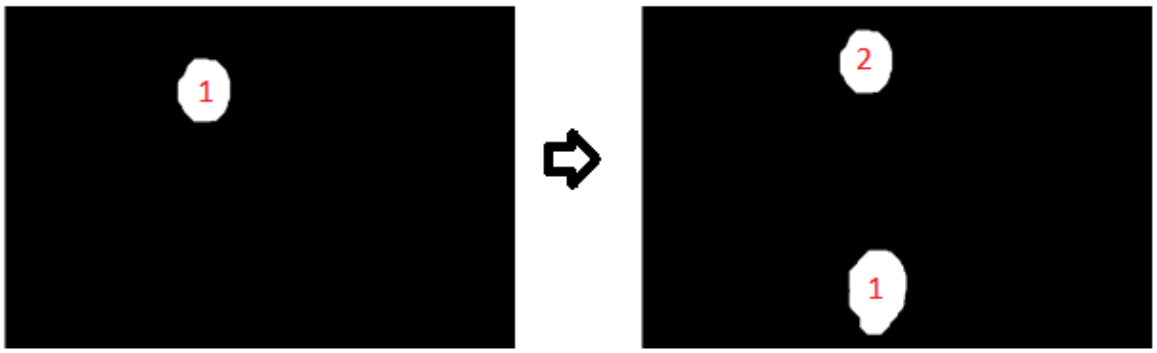

Figura 5.4. Cas d'aparició de la regió d'àrea major.

Com s'aprecia en la Figura 5.4, en aquest cas la regió que apareix en el nou fotograma correspon a la regió amb l'àrea més gran de les dues presents. Perquè això no afecti el bon funcionament de l'algorisme és necessari que les coordenades de la regió major del nou fotograma s'intercanviïn amb les de la regió menor. Només així s'estaran comprovant dues regions que equivalen al mateix objecte capturat i s'obtindran valors resultants vàlids.

Podrà observar-se que s'ha realitzat aquest intercanvi quan en la finestra de comandes s'hi mostri "Blob_Appearence_Case_2", tal com s'aprecia en el Fragment de codi 5.2.

```
if (((positions_imatge(2*t - 1) ~= 0)||(positions_imatge(2*t) ~=
0))&&(positions_imatge_update(2*t - 1) == 0)&&(positions_imatge_update(2*t) == 0))

        if (((area_update-(area_update*0.2)) <= area_gran)&&(area_gran <= area_update +
(area_update * 0.2))&&(apear == 0))
                disp('--------------->>Blob-Appearence_Case_1<<--------------')
                one_blob = 1;

        else
                positions_imatge_update(3) = positions_imatge_update(1);
                positions_imatge_update(4) = positions_imatge_update(2);
                positions_imatge_update(1) = pos_x_image_gran_update;
                positions_imatge_update(2) = pos_y_image_gran_update;
                disp('--------------->>Blob-Appearence_Case_2<<--------------')
                exep2 = 1;
                one_blob = 1;
                disapear = 1;
                cont_disapear = 0;
        end
else
        cont_apear = cont_apear + 1;
        if (cont_apear >= 4)
                apear = 0;
                cont_apear = 0;
        end
end
```

Fragment de codi 5.2. Detecció i actuació en la desaparició de regions.

5.1.1. Situacions en què els casos d'aparició i desaparició de regions són consecutives

Tant del Fragment de codi 5.1 com del Fragment de codi 5.2 pot apreciar-se que hi ha una part del codi que no s'ha comentat. Aquest equival a la situació que succeeix quan de manera consecutiva, una regió apareix i desapareix en fotogrames seguits tal com es mostra en la Figura 5.5.

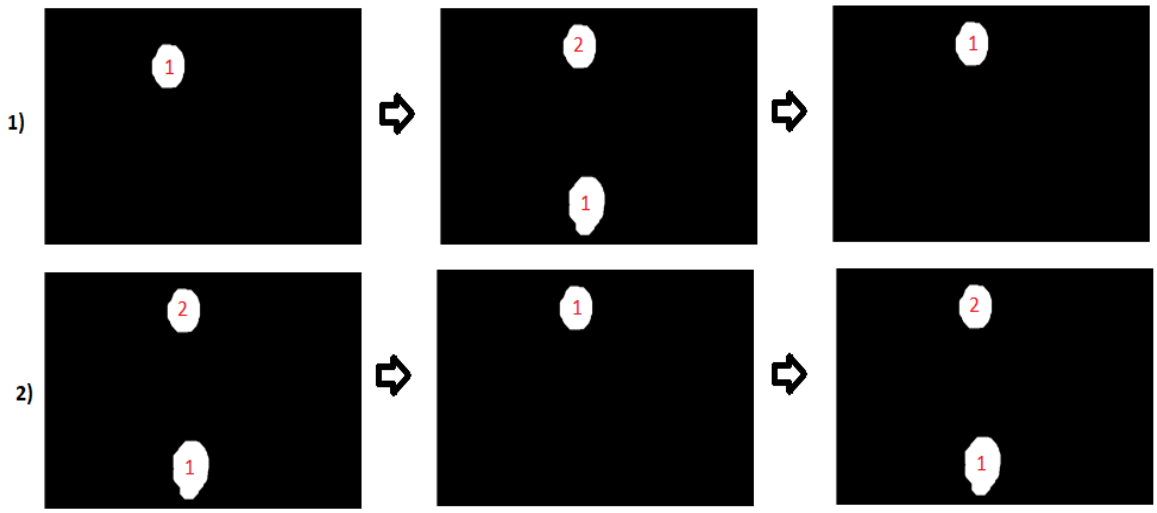

Figura 5.5. Casos en què les regions apareixen i desapareixen consecutivament.

En el cas de la fila 1 de la Figura 5.5 s'hi observa com de manera consecutiva hi ha un cas d'aparició d'una regió nova de major àrea a la ja coneguda i seguidament aquesta mateixa regió desapareix. Per tal de tractar aquesta situació, es creen les variables "appear" i "disappear". Quan es detecta que ha aparegut una regió d'àrea major a la del fotograma anterior la variable "appear" s'iguala a la unitat. Això fa que, quan en el següent fotograma se'n detecta la desaparició, el programa és capaç de seleccionar les coordenades correctes per tal de comparar posicions en l'eix de les x i y que corresponguin a la mateixa regió. A més a més, amb l'*else* que està assignat a la condició *if* general es torna a igualar a zero la variable "appear" que s'havia activat anteriorment quan en dos fotogrames consecutius no s'ha tornat a trobar la mateixa situació. Adonis el lector que en la condició encarregada de posar aquesta variable a zero està igualat a quatre i no a dos fotogrames perquè són dos les regions a processar i per tant aquesta condició es comprovarà dues vegades cada cicle.

Així mateix, en la segona fila de la Figura 5.5 s'hi aprecia el cas contrari que en la primera fila. En aquesta situació l'algorisme ha d'actuar correctament quan es detecti de manera consecutiva un cas de desaparició de la regió amb l'àrea major i tot seguit la reaparició d'aquesta. Se segueix la mateixa explicació que en el cas anterior amb la diferència que en aquest s'iguala a la unitat la variable "disappear".

5.2. Càlcul del desplaçament relatiu

Primer de tot del Fragment de codi 5.3 cal esmentar que és necessari actualitzar la variable de l'àrea. Així, en aquest projecte les variables que utilitzin el sufix "_update" en el nom de la variable es tracten com a la variable antiga d'alguna propietat o característica.

Seguint en l'anàlisi del mateix fragment de codi s'hi realitza el càlcul del desplaçament relatiu de fotograma a fotograma. Tanmateix, aquest queda condicionat per una estructura *If* que comprova que en efecte es restin les coordenades procedents sempre de la mateixa regió.

La condició *If* que s'aplica quan la variable "t" és igual a la unitat indicarà que s'està calculant la distància recorreguda per la regió amb una àrea major. Així mateix, la variable "one_blob" permetrà que es calculi la distància de l'única regió present en un fotograma en cas que així sigui. D'altra banda, la condició que comprova quan la variable "t" és igual a 2 indicarà que s'està calculant la distància recorreguda per la segona regió d'àrea major.

```
if (t == length(labels))
        area_update = area_gran;
end
if ((t == 1)||(one_blob == 1)) %The distance in pixels is found
        pos_x_image_res_gran  = positions_imatge(1) - positions_imatge_update(1);
        pos_y_image_res_gran = positions_imatge(2) - positions_imatge_update(2);
end
if (t == 2)
        pos_x_image_res_petita  = positions_imatge(3) - positions_imatge_update(3);
        pos_y_image_res_petita = positions_imatge(4) - positions_imatge_update(4);
        exep1 = 0;
end
```

Fragment de codi 5.3. Càlcul del desplaçament relatiu.

Cal que el lector prengui nota que el resultat del desplaçament relatiu de cadascuna de les regions es guarda en variables separades que més endavant es tractaran de manera convenient.

5.3. Conversió del desplaçament de píxels al sistema mètric

Havent obtingut ja el desplaçament relatiu realitzat pel robot entre dos fotogrames consecutius, és necessari convertir la distància obtinguda en píxels al sistema mètric, ja que aquesta serà la manera emprada per a referenciar el sistema a un plànol d'un habitatge així com comparar el sistema desenvolupat en aquest projecte amb altres sistemes. Per tal de realitzar tal conversió, s'utilitza la tècnica que es desenvolupa en la referència [1]. Tanmateix, en aquesta solament es tenen en compte els desplaçaments que es realitzen en l'eix d'abscisses.

Primer de tot caldrà calcular la cobertura a una distància particular:

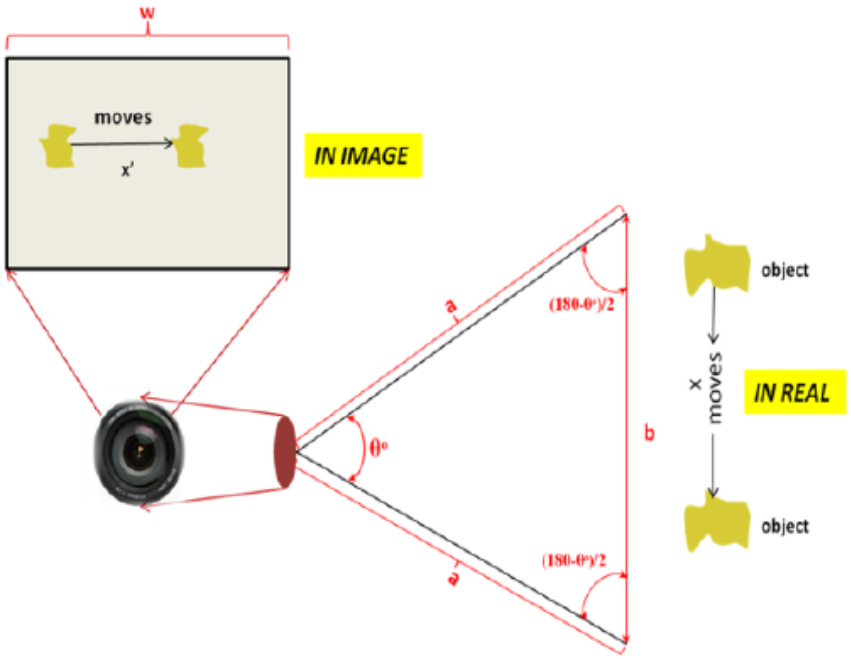

Figura 5.6. Tècnica per obtenir el desplaçament real en l'eix d'abscisses (font [1]).

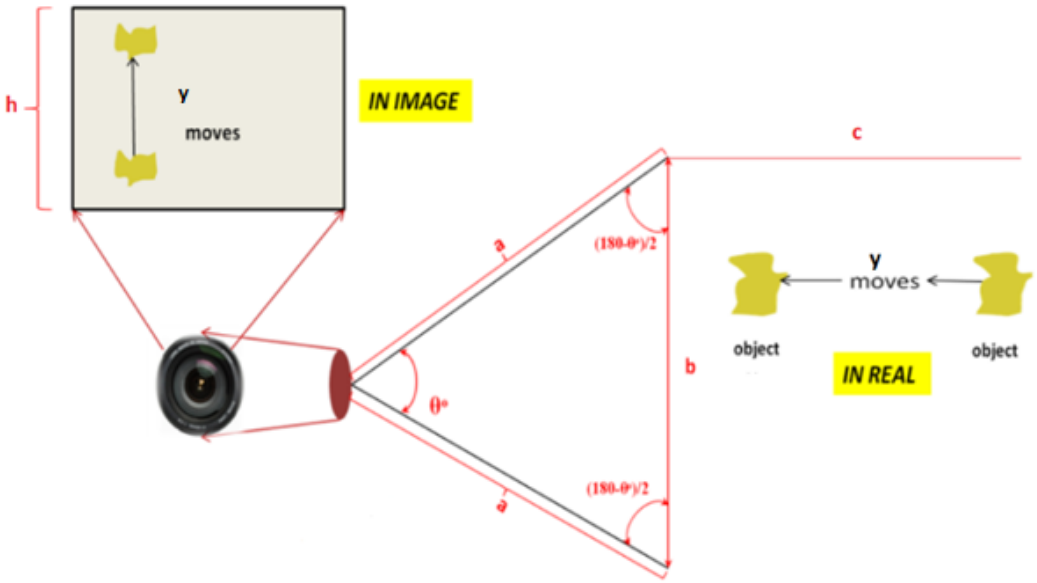

Figura 5.7. Tècnica per obtenir el desplaçament real en l'eix d'ordenades.

Per tal de calcular el desplaçament real del robot primer cal conèixer les propietats de la càmera que s'utilitzarà per obtenir les imatges que es processaran. Observant les imatges de la Figura 5.6 i la Figura 5.7 s'hi observa que utilitzant trigonometria és possible obtenir les equacions necessàries. Així mateix,

coneixent la distància cap al sostre "a" i l'angle de visió de la càmera "θ" o FOV (Field Of View) és possible trobar la incògnita "b" i "c".

De la Figura 5.6 se'n pot obtenir la següent equació:

$$\frac{b}{sin\theta} = \frac{a}{\sin(\frac{(180-\theta)}{2})} \quad \textbf{(Eq. 5.1)}$$

$$b = \frac{a * sin\theta}{\sin(\frac{(90-\theta)}{2})} \quad \textbf{(Eq. 5.2)}$$

L'equació anterior pot ser simplificada a:

$$b = 2 * a * \sin(\frac{\theta}{2}) \quad \textbf{(Eq. 5.3)}$$

D'igual manera, de la Figura 5.7 se'n pot obtenir la següent equació:

$$\frac{c}{cos\theta} = \frac{a}{\cos(\frac{(180-\theta)}{2})} \quad \textbf{(Eq. 5.4)}$$

$$c = \frac{a * cos\theta}{\cos(\frac{(90-\theta)}{2})} \quad \textbf{(Eq. 5.5)}$$

L'equació anterior pot ser simplificada a:

$$c = a * cos(\frac{\theta}{2}) \quad \textbf{(Eq. 5.6)}$$

Seguidament es calcularà el moviment de la regió associada l'objecte. Coneixent "b" i "c" amb les expressions de les equacions 5.3 i 5.6 és possible mesurar el moviment d'una regió a una distància "a" simplement amb les imatges capturades. A més a més, també és possible conèixer la quantitat de píxels que la imatge tindrà en horitzontal ("w") i en vertical ("h"). Tal com es planteja en la referència [1], sabent que el valor d'aquestes dues variables correspon al valor de l'àrea que es pot cobrir en qualsevol moment es pot deduir que, coneixent la distància "a", el moviment d'un píxel de la regió en la imatge equivaldrà a:

$$x = \frac{b}{w} * x' \qquad \textbf{(Eq. 5.7)}$$

$$y = \frac{c}{h} * y' \qquad \textbf{(Eq. 5.8)}$$

```
if (t == 2) %check regions and relative displacement
        if ((first_blob == 1)&&(cont_first_blob == 0))
                disp('First blob with x=0cm and y=0cm')
                cont_first_blob = 1;
        elseif ((((positions_imatge(2*t - 1) == 0)&&(positions_imatge(2*t) ==
0))||one_blob == 1)&&(first_blob) == 1)
                if (exep2 == 1)
                        pos_x_real = pos_x_image_res_petita * image2real_x;
                        pos_y_real = pos_y_image_res_petita * image2real_y;
                        exep2 = 0;
                        one_blob = 0;
                        angle_res180 = 0;
                else
                        pos_x_real = pos_x_image_res_gran * image2real_x;
                        pos_y_real = pos_y_image_res_gran * image2real_y;
                        one_blob = 0;
                        angle_res180 = 0;
                end
        elseif ((positions_imatge(2*t - 1) ~= 0)&&(positions_imatge(2*t) ~=
0)&&(first_blob == 1))
                if (((-5 <= pos_x_image_res_gran)&&(pos_x_image_res_gran <= 5))||((-5 <=
pos_x_image_res_petita)&&(pos_x_image_res_petita <= 5)))
                        pos_x_real = 0;
                else
                        pos_x_real = ((pos_x_image_res_gran + pos_x_image_res_petita)/2)
* image2real_x;
                end
                if (((-5 <= pos_y_image_res_gran)&&(pos_y_image_res_gran <= 5))||((-5 <=
pos_y_image_res_petita)&&(pos_y_image_res_petita <= 5)))
                        pos_y_real = 0;
                else
                        pos_y_real = ((pos_y_image_res_gran + pos_y_image_res_petita)/2)
* image2real_y;
                end
        end
```

Fragment de codi 5.4. Càlcul del desplaçament relatiu real.

5.3.1. Càlcul del desplaçament relatiu real

Tal com es pot apreciar en el Fragment de codi 5.4 i en la Figura 5.8, la primera de les vegades que l'aplicació es disposa a executar aquest fragment de codi, aquest entra en la primera de les condicions *If* gràcies a la variable "first_blob". En aquesta no computa el desplaçament relatiu, ja que al ser la primera de les comparacions les variables equivalents a característiques de fotogrames anteriors encara no s'han inicialitzat a valors correctes i dóna desplaçaments inexactes. Per tant, simplement es mostra que és la primera de les regions computades amb un desplaçament relatiu de zero en els dos eixos. A més a més, també s'incrementa el valor d'un comptador intern perquè aquesta condició no torni a complir-se.

En la segona de les condicions, ara *elseif,* es comprova si les coordenades de la regió de menor àrea són iguals a zero. En aquest cas es confirmaria que només hi ha una regió en el fotograma. Aquest cas també podrà confirmar-se si alguna de les situacions de la Figura 5.1 o Figura 5.2 ocorre gràcies a la variable "one_blob". Òbviament, la variable "first_blob" emprada anteriorment ha d'haver estat activada prèviament. Quan ja s'ha complert aquesta última condició es realitza la conversió de píxels de la imatge al sistema mètric de la regió que compleixi la condició "exep2 ==1" prèviament establerta.

Per últim es comprova la condició que hi hagi dues regions en el fotograma. Aquest cas es compararà que les coordenades de la regió amb l'àrea menor siguin diferents de zero, ja que en aquesta situació es pot confirmar que hi ha dues regions en el fotograma. A més d'aquesta condició també serà necessari que, l'igual que l'anterior, la variable "first_blob" hagi estat activada anteriorment. Quan aquestes condicions es compleixen és possible calcular el desplaçament mitjà en píxels realitzat pel robot i convertir-lo en desplaçament real. El desplaçament mitjà es calcularà mitjançant el desplaçament realitzat per la regió d'àrea major i el desplaçament realitzat per l'àrea menor. Tanmateix, per tal de calcular aquest valor hauran de superar el filtre que s'aplica a tots aquells valors que tenen valors inferiors als que s'estimen de ser significatius i que per tant, es consideren soroll del sistema.

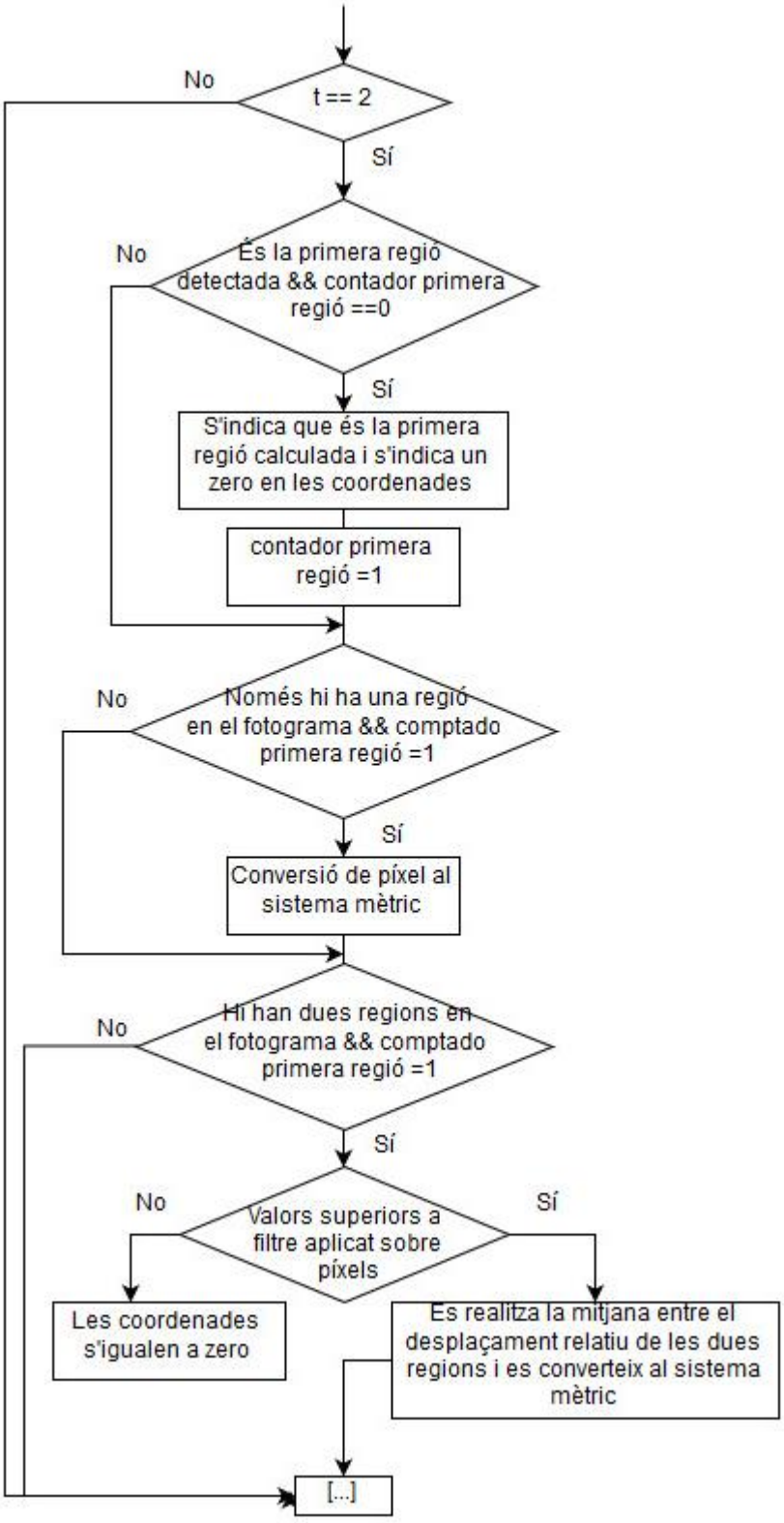

Figura 5.8. Diagrama de blocs del funcionament del sistema de conversió de píxels al sistema mètric.

5.4. Càlcul dels angles relatius

5.4.1. Càlcul de l'angle relatiu mitjançant dues regions

En el Fragment de codi 5.5 es realitzen les operacions que es mostren gràficament en les imatges de la Figura 5.9 i Figura 5.10. Analitzant el fragment de codi des del principi pot apreciar-se com la clau del funcionament del càlcul de l'angle utilitzant fotogrames a on apareixen dues regions recau en què aquest serà relatiu. Això significa que tal com s'observa en la Figura 5.10 el càlcul de l'angle mitjançant dues regions és resultat de la diferència entre dos fotogrames consecutius.

```
if ((area_petita ~= 0)&&(t == length(labels)))
        mini_counter = mini_counter + 1;
        u= [(positions_imatge(1)-positions_imatge(3)) (positions_imatge(2)-
positions_imatge(4)) 0];
        v= [0 1 0];
        angle180 =  atan2d(norm(cross(u,v)),dot(u,v));
        if (mini_counter >= 4)
                angle_res180 = (angle180 - angle_update180);
        end
        angle_update180 = angle180;
elseif (area_petita == 0)
        mini_counter = 0;
        angle_update180 = 0;
end
fprintf('>Between two blobs the robot moved %fdegrees\n',accu_angle180_res)
```

Fragment de codi 5.5. Càlcul de l'angle relatiu amb dos regions.

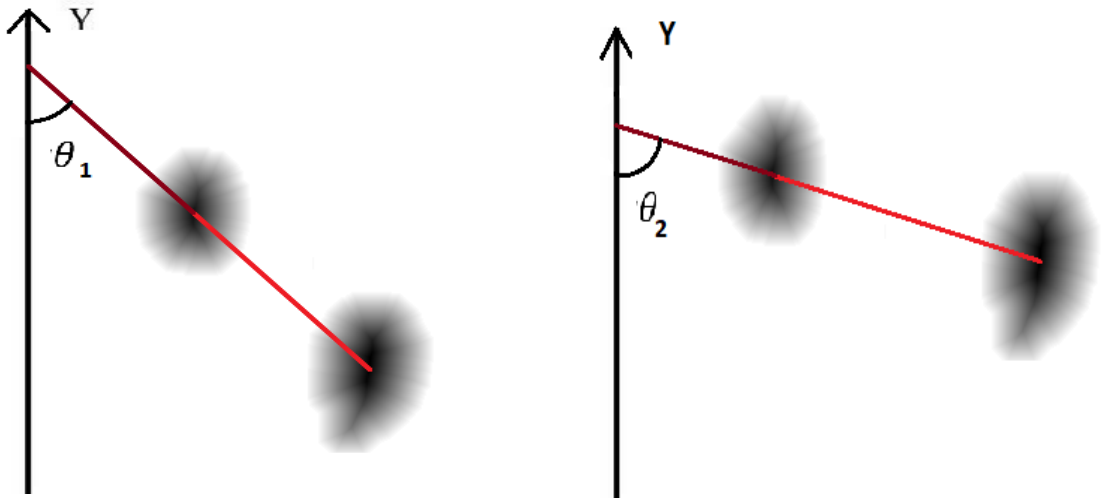

Figura 5.9. Càlcul de l'angle format pel vector que uneix dues regions amb l'eix d'ordenades en dos fotogrames diferents.

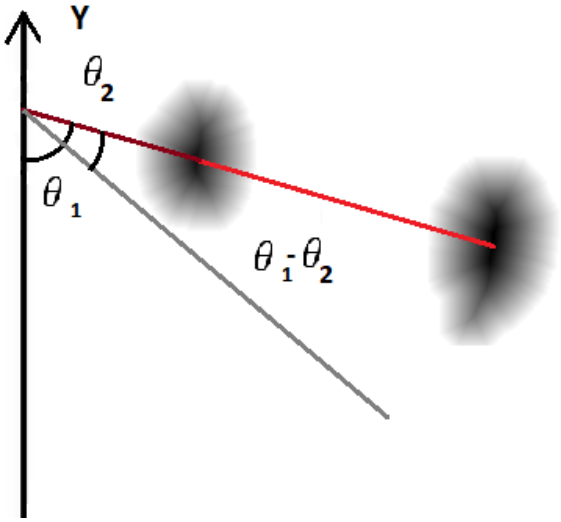

Figura 5.10. Càlcul de l'angle relatiu entre dos fotogrames.

Com ja s'ha comentat en el transcurs d'aquest projecte, la variable assignada amb el sufix "_update180" correspon l'angle θ_1 de la Figura 5.9. Així doncs, l'angle θ_2 correspon a la variable "angle_180" del fragment de codi que ens ocupa. S'aprecia en aquest que en el cas que es detecti que no hi ha dues regions en el fotograma que s'està processant aleshores es reinicien les variables que fins al moment s'estaven emprant per al càlcul de l'angle.

Aquest algorisme per al càlcul de l'angle mitjançant dues regions s'emprarà per al càlcul d'aquelles situacions en les quals el robot realitzi un canvi de direcció de 180º. Aquesta és una de les situacions més complicades de detectar en el cas que es dugui a terme sense canviar de posició i sobre el mateix robot, ja que aleshores no existeix un desplaçament relatiu de la regió entre dos fotogrames consecutius. En el cas que aquesta succeeixi quan el sistema a detectat dues regions en el fotograma es podrà detectar la situació ràpidament mitjançant l'algorisme ja explicat.

5.4.2. Càlcul de d'angle relatiu mitjançant una regió

En el cas que el sistema només detecti una regió en el fotograma, el càlcul de l'angle relatiu es realitzarà mitjançant l'algorisme del Fragment de codi 5.6. La idea principal d'aquest algorisme queda plasmada en la Figura 5.11. L'algorisme en aquest cas crearà un vector que utilitzarà el desplaçament relatiu real com a punt final del vector que tindrà per inici l'origen de coordenades. Així, variant l'eix sobre el qual es calcula aquest angle segons el signe dels paràmetres que conformen el desplaçament relatiu és possible calcular l'angle θ que es mostra en la figura que ens ocupa. La variable encarregada de realitzar els canvis de l'eix sobre el qual es calcularà l'angle, "angle_axis", es controla més endavant i l'explicació del funcionament de la mateixa es troba en l'apartat 6.1.

```
u= [(pos_x_real) (pos_y_real) 0];
if (angle_axis == 1)
        if (pos_x_real >= 0)
                v= [1 0 0];
         elseif (pos_x_real < 0)
                v= [-1 0 0];
         end
elseif (angle_axis == 0)
        if (pos_y_real >= 0)
                v= [0 1 0];
         elseif (pos_y_real <0)
                v= [0 -1 0];
         end
end
angle=  atan2d(norm(cross(u,v)),dot(u,v));
fprintf('Computed %fdegrees\n',angle)
```

Fragment de codi 5.6. Càlcul de l'angle relatiu.

Així com el mètode que s'ha explicat anteriorment s'utilitza pel càlcul d'aquelles situacions que el robot du a terme un canvi de direcció de 180º quan hi hagi com a mínim dues regions en el fotograma que es processa, aquest algorisme s'utilitzarà per al càlcul d'aquelles situacions que indiquin un gir de 90º tant sigui aquest positiu com negatiu. Tanmateix, també és possible utilitzar aquest algorisme per a detectar aquelles situacions en les quals el robot du a terme un canvi de sentit de 180º amb només una regió activa en el fotograma que es processa aconseguint que aquest detecti dues vegades consecutives un gir de 90º del mateix sentit.

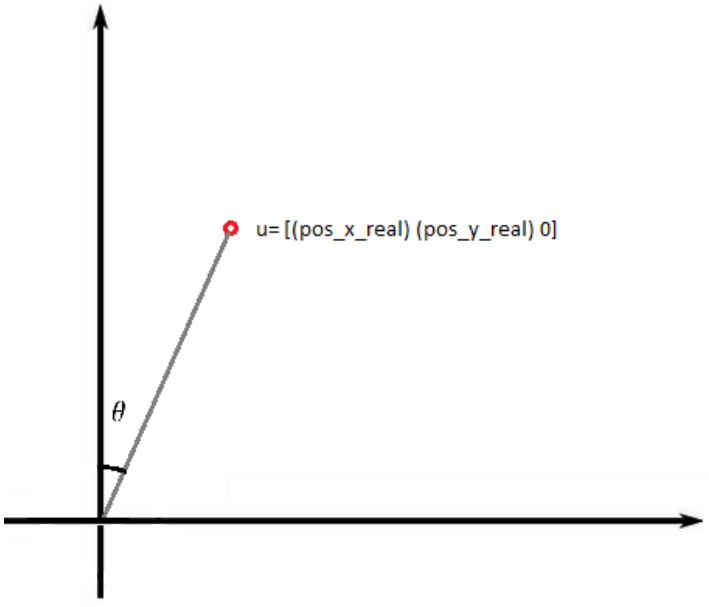

Figura 5.11. Càlcul de l'angle relatiu mitjançant quan hi ha una sola regió en el fotograma.

6. Resultats

6.1. Algorisme de traçada del recorregut

Per tal de traçar correctament en una gràfica la trajectòria del robot és necessari tractar els valors del desplaçament i angle prèviament obtinguts. La raó principal és que mitjançant els algorismes que s'han explicat en el transcurs d'aquest projecte no és possible traçar la trajectòria tal com caldria esperar, ja que el sistema no fusioni les dades del desplaçament i angle relatius. Per tant d'obtenir la direcció del robot és fonamental tractar els angles relatius de la manera adequada per tal de, així, canviar a conveniència els paràmetres de les coordenades que conformen la gràfica de la trajectòria.

Aquesta explicació la complementen el diagrama de blocs de la Figura 6.1 així com el Fragment de codi 6.1 on s'adjunten les línies de codi més significatives en les operacions que es clarificaran en aquest apartat.

Seguint l'ordre natural del diagrama de blocs, s'observa com abans de traçar la trajectòria es comprova que el desplaçament relatiu calculat sigui d'un valor plausible. Per a fer aquesta comprovació s'empra una condició *If* que comprova que el valor desplaçat no sigui major del 25% de la mida del fotograma convertida al sistema mètric.

A continuació, en cas d'haver superat el filtre anterior se n'aplica un altre que en aquest cas s'encarrega de filtrar tots aquells valors que estiguin entre ±2cm. Aquest s'aplica perquè es considera que quan el desplaçament calculat estigui entre aquests valors és susceptible d'haver sigut produït per inexactituds del sistema. Les possibles causes d'inexactituds es tractaran més endavant quan s'analitzin els resultats obtinguts.

Tot seguit, continuant en l'anàlisi dels processos necessaris per traçar la trajectòria en una gràfica, es comprova l'estat de la variable "accu_angle180_res". Aquesta variable equival a la suma del valor resultant de l'algorisme que calcula l'angle relatiu quan hi ha com a mínim dues regions en el fotograma. Aquesta suma es durà a terme entre aquells fotogrames consecutius que en els que es detecti un angle relatiu major de zero. Quan el resultat de l'algorisme que calcula l'angle, Fragment de codi 5.5, sigui zero la variable "accu_angle180_res" també es reiniciarà a zero. Així doncs, quan es compleixi la condició que la variable "accu_angle180_res" té un valor major de 170º es considerarà que s'ha dut a terme un canvi de direcció de 180º i es tractaran les variables de l'eix d'ordenades en conseqüència.

Cal tindre en compte que, per tal de dur a terme una gràfica que mostri la trajectòria del robot, els valors de les coordenades van sumant-se successivament i que, per tant, l'error del sistema s'anirà acumulant amb el temps.

Seguidament, es comprovava l'estat de les variables "angle_exception_right" i "angle_exception_left". Aquestes defineixen les accions que cal dur a terme en cas que el robot realitzi girs de 90º, sigui quin sigui el seu sentit. Tal com s'observa en el Fragment de codi 6.1, en cas que es compleixin les condicions que s'hi estableixen es modificaran en conseqüència les coordenades en qüestió. Tanmateix, tal com s'hi aprecia, les variables que ens ocupen han d'haver sigut prèviament establertes. Per tal de definir-les s'empra un mètode similar a l'explicat anteriorment per l'angle calculat en aquells fotogrames on hi ha dues regions. Així doncs cada vegada que la variable "accu_angle_res", equivalent a la "accu_angle180_res" tractada anteriorment, tingui un valor superior a 91º o inferior a -91º, s'aplicarà un canvi de sentit que afectarà directament a les variables "angle_exception_left" i "angle_exception_right", directament responsables de les operacions que s'aplicaran a les coordenades que traçaran la trajectòria.

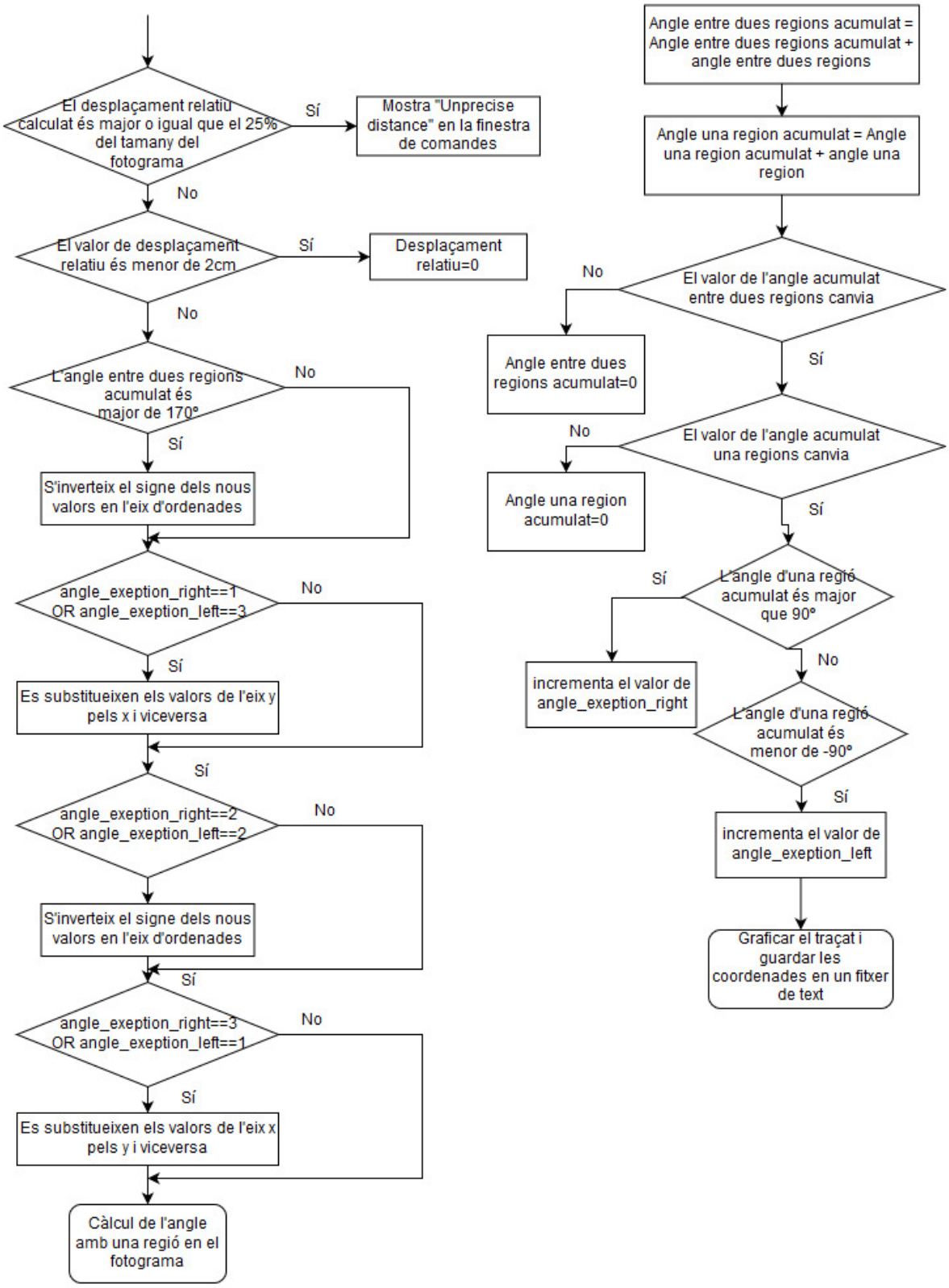

Figura 6.1. Diagrama de blocs del funcionament de l'algorisme de traçada del recorregut.

```
if (((accu_angle180_res >= 65)&&(pos_y_real_reverse >= 8))||(angle180_exception == 1))
        disp('---->CHANGE 180°<----');
        pos_y_real = -pos_y_real;
        if ((angle180_exception == 0) && (pos_y_real_reverse >= 8))
                angle180_exception = 1;
        elseif ((angle180_exception == 1) && (pos_y_real_reverse >= 8))
                angle180_exception = 0;
        end
        pos_y_real_reverse = 0;
end
if ((angle_exception_right == 1)||(angle_exception_left == 3))
        disp('---->CHANGE_RIGHT<----');
        pos_y_real_mom= pos_y_real;
        pos_y_real = pos_x_real;
        pos_x_real = pos_y_real_mom;
        angle_axis = 1;
        angle_cont = angle_cont + 1;
elseif ((angle_exception_right == 2)||(angle_exception_left == 2))
        disp('---->CHANGE_180<----');
        pos_y_real = -pos_y_real;
        angle_axis = 0;
elseif ((angle_exception_right == 3)||(angle_exception_left == 1))
        disp('---->CHANGE_LEFT<----');
        pos_y_real_mom = pos_y_real;
        pos_y_real = -pos_x_real;
        pos_x_real = -pos_y_real_mom;
        angle_axis = 1;
end
→Càlcul de l'angle amb una regió en el fotograma
if (angle_axis == 1)
        if (pos_y_real < 0)
                angle = -angle;
        end
elseif (angle_axis == 0)
        if (pos_x_real < 0)
        angle = -angle;
        end
end
accu_pos_x_real=accu_pos_x_real + pos_x_real;
accu_pos_y_real=accu_pos_y_real + pos_y_real;
accu_angle_res = accu_angle_res+angle;
accu_angle180_res = accu_angle180_res+angle_res180;
if (accu_angle180_res == accu_angle180_res_update)
        accu_angle180_res=0;
else
        pos_y_real_reverse = pos_y_real_reverse +1;
end
if (accu_angle_res == accu_angle_res_update)
        accu_angle_res=0;
end
accu_angle_res_update = accu_angle_res;
accu_angle180_res_update = accu_angle180_res;
if ((accu_angle_res >= 91)&&(angle >= 4)) %91
        if (angle_exception_left == 0)
                if (angle_exception_right < 4)
                        angle_exception_right = angle_exception_right+1
                end
        else
                angle_exception_left = angle_exception_left-1
        end
elseif ((accu_angle_res <= -91)&&(angle <= -4))
        if (angle_exception_right == 0)
                if (angle_exception_left < 4)
                        angle_exception_left = angle_exception_left+1
                end
         else
                angle_exception_right = angle_exception_right-1
        end
end
```

Fragment de codi 6.1. Operacions necessàries per a traçar la trajectòria del robot correctament.

6.2. Resultats experimentals en un habitatge convencional

Per provar l'efectivitat del codi desenvolupat es prepara el sistema que apareix en la Figura 6.2. Amb aquest es gravaran els vídeos que més endavant seran processats per l'algorisme.

Figura 6.2. Eixos aplicats al robot utilitzat per les proves experimentals en un habitatge.

Les proves que es realitzen en aquest projecte en un entorn d'un habitatge convencional poden veure's gràficament resumides en la Figura 6.3. Cada una de les trajectòries equival a un color diferent. A més a més, cal esmentar que en aquestes, el punt inicial queda descrit amb el número 1 i el punt final amb el 2.

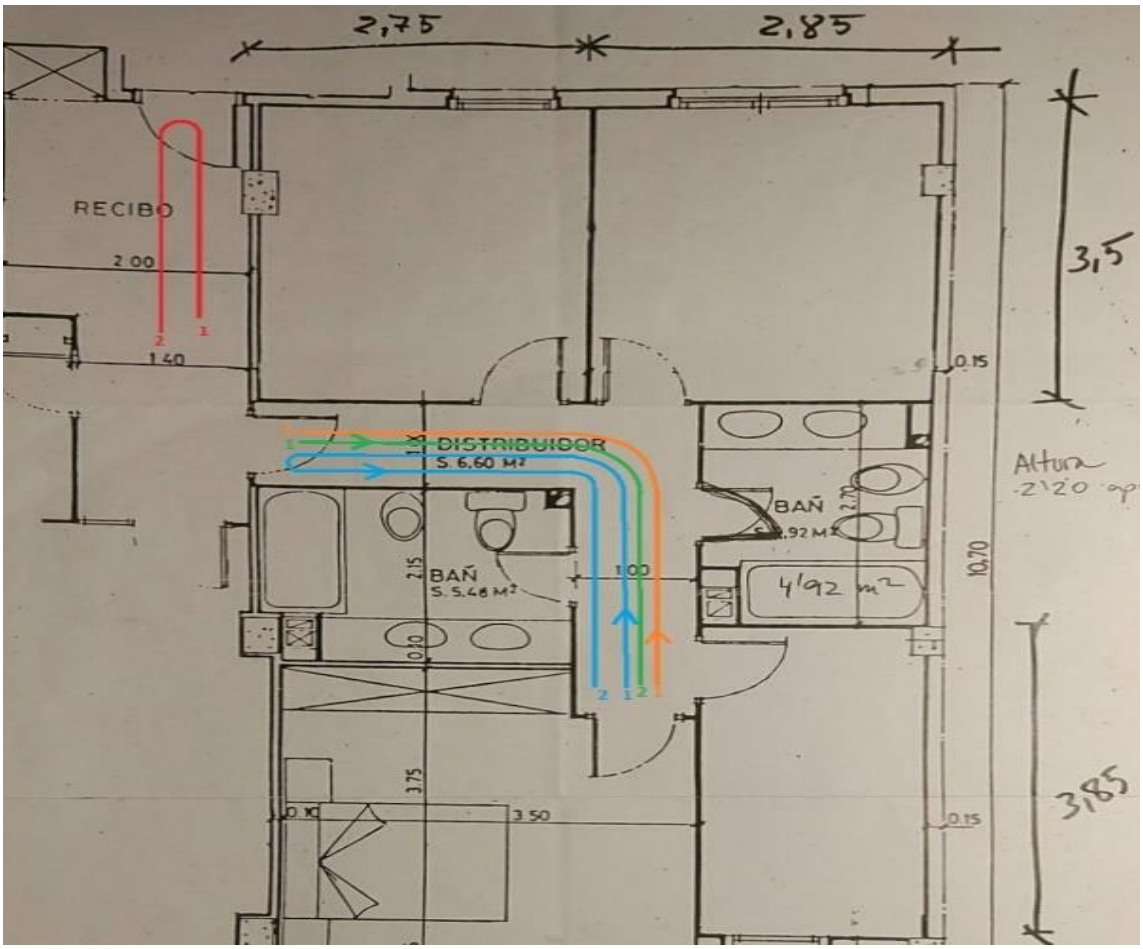

Figura 6.3. Plànol de l'habitatge amb els recorreguts on s'ha provat el programa.

Com a primer test del programa desenvolupat es decideix provar la capacitat de mesurar la distància recorreguda pel robot així com la capacitat de detectar situacions en les quals és necessari que realitzi un gir de sentit de 180º en la traçada de la trajectòria.

Pel que fa a la definició dels paràmetres de la distància de la càmera al sostre aquesta equivaldrà a 240 centímetres i l'angle de visió o FOV es definirà a 60º. Aquests valors seran certs per totes aquelles proves que es realitzen en l'habitatge convencional d'aquest projecte. A més a més, es defineix el llindar de binarització a 0,95 així com l'amplada del fotograma a 720 píxels.

D'altra banda, cal apuntar que en les traçades dels resultats l'inici de la trajectòria sempre serà l'inici de coordenades (0,0).

Es comença doncs per la trajectòria vermella de la Figura 6.3. En aquesta es recorre una trajectòria teòrica de 540 cm en total així com un canvi de direcció de 180º. Tal com s'aprecia en el resultat de l'algorisme de la Figura 6.4, el programa ha sigut capaç de detectar el canvi de direcció en el moment adequat així com de mesurar la distància recorreguda amb bastant exactitud.

En aquesta primera prova el canvi de direcció va dur-se a terme en un moment en el qual hi havia dues regions actives en el fotograma per tal de provar l'efectivitat del Fragment de codi 5.5.

S'aprecia com l'algorisme en qüestió a funcionat de manera adequada tot i que cal esmentar que la distància total recorreguda que figura en el resultat del programa és aproximadament 35 cm més curta que la distància teòricament recorreguda. La raó per aquesta imprecisió és un dels majors impediments del sistema. Aquest és que quan el programa no detecta cap regió en el fotograma no és capaç de discernir el moviment relatiu del robot respecte a l'entorn perquè no té cap element de l'entorn des del qual pugui referenciar-se.

D'altra banda, també s'hi aprecien lleugeres desviacions en l'eix d'abscisses que idealment no haurien de ser-hi. Tanmateix, difícilment desmereixen el resultat total, ja que tal com apreciar-se en l'escala del mateix eix, aquesta és una desviació molt petita (menys de 10 cm acumulats).

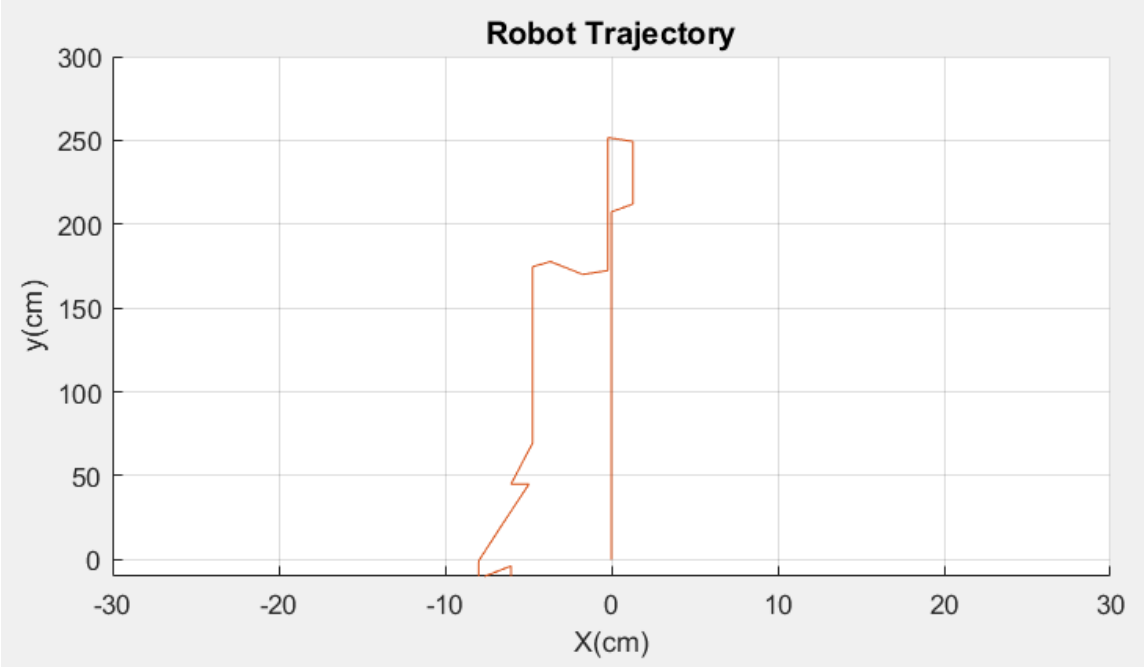

Figura 6.4. Traçat de la trajectòria vermella.

A continuació es realitza la prova de la trajectòria que segueix la línia verda de la Figura 6.3. En aquesta es comprova l'efectivitat del Fragment de codi 5.6 i del Fragment de codi 6.1.

Es comprova en la Figura 6.5 com el codi realitza el canvi de sentit de 90º de manera adequada. A més a més, també s'aprecia com en aquesta prova hi ha molt poques interferències pel que fa a desplaçaments no desitjats i s'aconsegueixen perfectament rectes que reflecteixen amb exactitud el moviment del robot.

En aquesta prova, el desplaçament teòric abans de realitzar el canvi de sentit de 90º era d'un total de 335 cm aproximadament. Tanmateix, s'aprecia com en la traçada de la trajectòria de la gràfica que s'està tractant el canvi es realitza aproximadament quan el robot ha recorregut una distància de 410 cm. La raó per aquesta inexactitud és que la variable "accu_angle_res" assoleix el valor llindar per a realitzar el canvi de direcció lleugerament més tard del que caldria esperar.

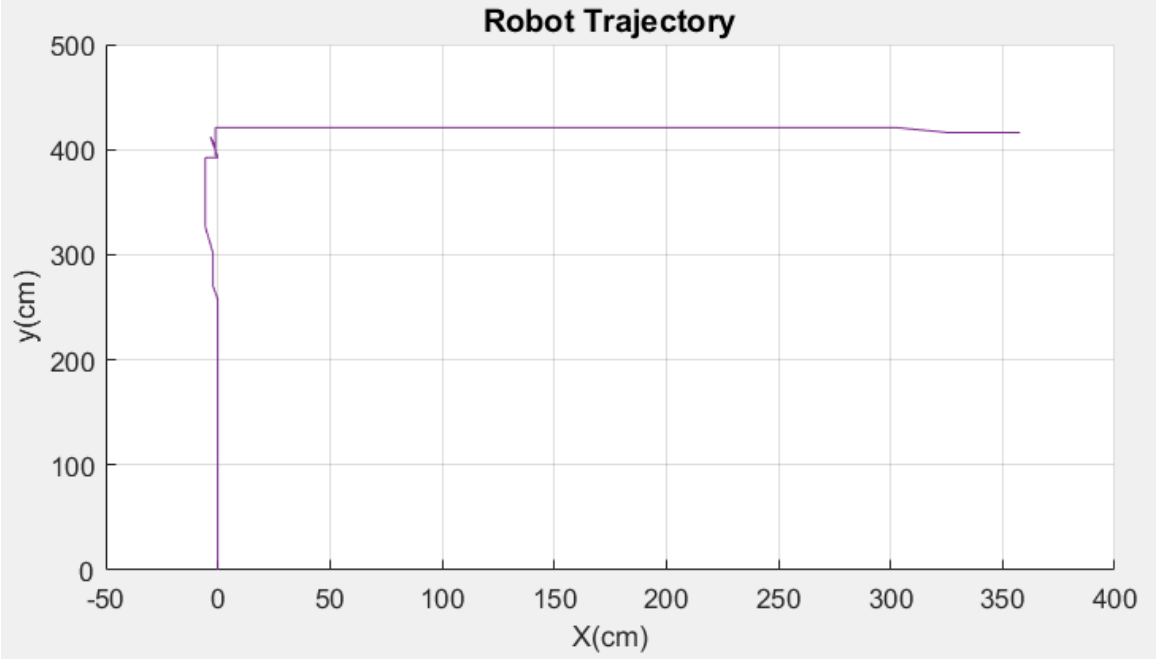

Figura 6.3

Figura 6.5. Traçat de la trajectòria verda.

Seguidament es realitza la prova de la trajectòria que segueix la línia taronja de la Figura 6.3. En aquesta es comprova l'efectivitat del Fragment de codi 5.6 i del Fragment de codi 6.1.

En aquest cas, s'aprecia en la Figura 6.6 com es tracta d'un cas molt similar a l'anterior amb la lleugera diferència que en aquest és un canvi de sentit de -90º. S'observa a més a més que, igual que en el cas anterior, també hi apareix un cert retard en la detecció d'aquesta situació fet que fa que també hi hagi lleugeres inexactituds en la mesura de la distància recorreguda.

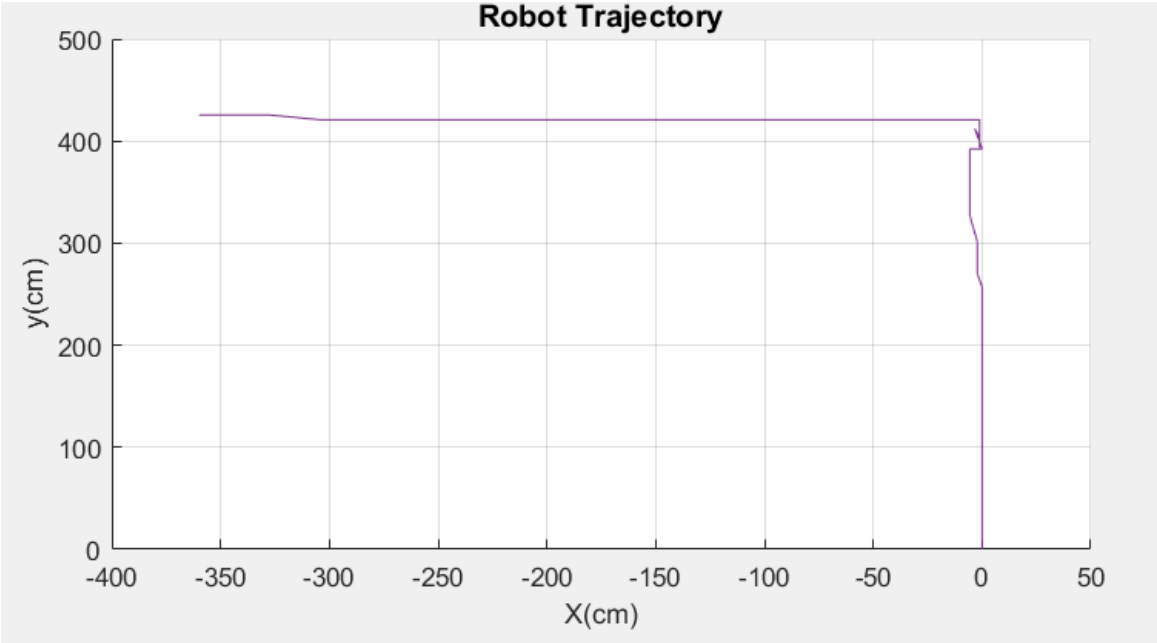

Figura 6.6. Traçat de la trajectòria taronja.

Aquesta es l'última de les proves que es realitzen en l'entorn d'un habitatge convencional per provar la viabilitat del sistema. En aquest cas és el robot realitza la trajectòria marcada per la línia blava de la Figura 6.3. Aquesta és una de les situacions més complexes que el sistema ha de ser capaç de detectar, ja que es conjuren les situacions esmentades anteriorment referents al funcionament del Fragment de codi 5.6 i del Fragment de codi 6.1.

A més a més, i a diferència dels casos esmentats anteriorment, en aquest cas el sistema tractarà amb un canvi de direcció de 180º en el qual només hi ha una regió en el fotograma. Això significa que no podrà utilitzar-se el Fragment de codi 5.5, dissenyat especialment per aquestes ocasions, sinó que es detectaran de manera consecutiva dos canvis de direcció de 90º en el mateix sentit.

Com pot apreciar-se en la Figura 6.7, és la detecció del canvi de sentit quan només hi ha una regió disponible en el fotograma la que causa més dificultats al sistema. S'observa com, tot i que la distància total que recorre el robot fins a realitzar el canvi de sentit és correcta, hi ha fluctuacions destacables en el sistema provinents dels canvis que s'apliquen a les coordenades quan es realitza un canvi de sentit de ±90º.

D'altra banda, s'aprecia en la figura en qüestió com les trajectòries traçades en la gràfica referents al camí d'anada i tornada d'ençà que es realitza en canvi de sentit de 180º difereixen en menys de 50 cm. Tenint en compte que la càmera no està posicionada al centre del robot i que el cos del robot en qüestió té un diàmetre de 33 cm és un error assumible provinent de les condicions en les quals s'han dut a terme els assajos.

Finalment, per finalitzar l'anàlisi d'aquesta trajectòria, s'aprecia com la distància que es detecta pel que fa al camí de tornada és clarament inferior a la distància que el robot s'ha desplaçat en la realitat, ja que la coordenada inicial hauria de concordar amb la coordenada final. S'observa que no és el cas. La raó principal per la qual el sistema no ha sigut capaç de traçar la trajectòria correctament és que l'error provinent dels girs de 90º i del de 180º provoquen que el sistema acumuli més error de l'esperat.

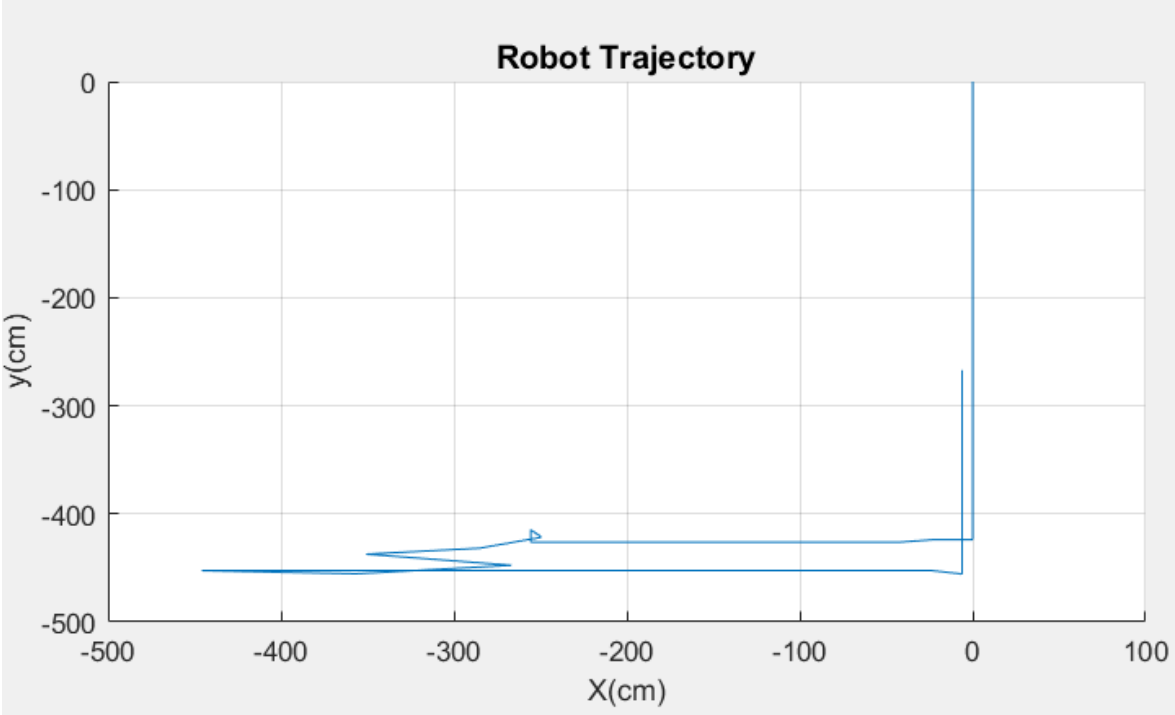

Figura 6.7. Traçat de la trajectòria blava.

En la Figura 6.8 s'hi aprecia el *workspace* que es crea per tal de mostrar l'usuari que executa el programa utilitzant com a entrada un vídeo pugui observar el traçat de la trajectòria del robot així com el processament de cada fotograma en els processos més crítics.

- En el número 1 d'aquesta figura s'hi mostra el fotograma sense processar extret directament del vídeo que s'utilitza com a entrada del programa. A més a més, just a sobre d'aquesta imatge s'hi mostra el nombre de fotogrames que s'han processat així com el nombre total de fotogrames que conformen el vídeo a tractar.
- En el número 2 de la figura s'hi mostra la imatge binaritzada amb el llindar manual després d'haver aplicat l'operació d'obertura i abans d'haver eliminat aquelles regions que puguin estar en contacte amb les cantonades del fotograma.
- En el número 3 de la figura hi apareix el fotograma després que s'hagi aplicat la transformada de distàncies.
- Per últim, en el número 4 hi apareix el traçat en temps real de la trajectòria realitzada pel robot mòbil.

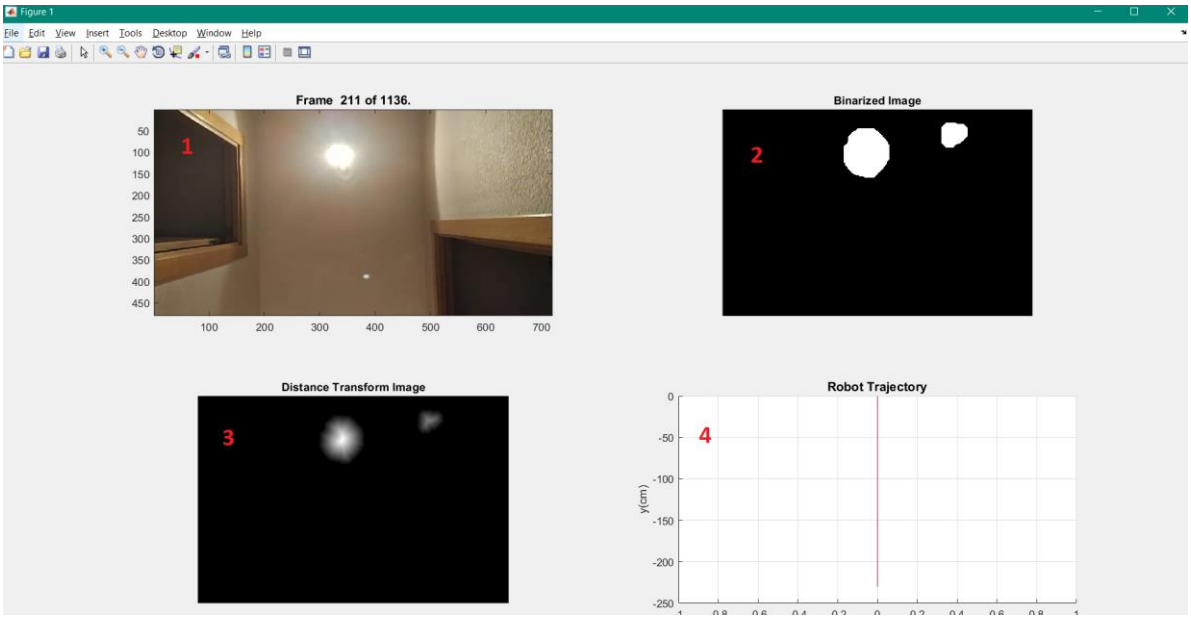

Figura 6.8. *Workspace* creat en el qual es mostra els processos més rellevants en el processament dels fotogrames.

7. Comparació del resultat amb altres odometries

Per tal de discernir la precisió del sistema desenvolupat en aquest projecte es comparen els resultats amb dos altres sistemes d'odometria. Per l'elecció de les odometries amb les que es realitzarà l'esmentada comparació es tenen en compte factors com el preu total del sistema o la precisió d'aquest.

Així doncs, els dos sistemes alternatius utilitzats per a la comparació són odometria mitjançant codificadors col·locats a les rodes i odometria mitjanant ultraviolada.

- Odometria mitjançant codificadors:

Es decideix emprar aquest tipus d'odometria perquè és una de les tècniques més esteses en la pràctica de la localització de robots mòbils. Tal com s'aprecia en la Figura 7.1, s'utilitza el Robot Pioneer 3-AT i els seus codificadors per tal d'obtenir l'odometria desitjada.

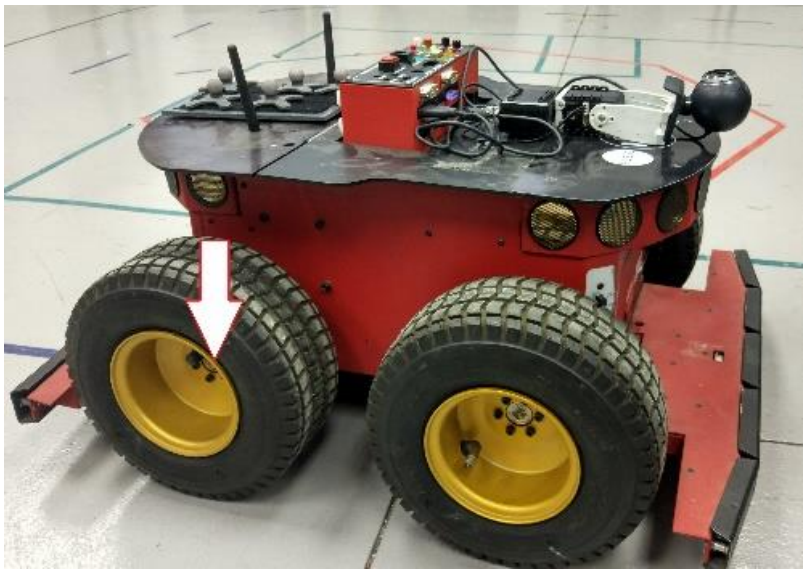

Figura 7.1. Fotografia del robot Pioneer 3-AT on es s'assenyala un dels dos codificadors.

Les rodes d'aquest robot mòbil tenen un diàmetre de 220 mm i els codificadors d'aquest tenen una precisió de 138 polsos/mm tal com s'esmenta en la referència [6]. Sabent que cada revolució de la roda són 500 polsos obtenim que la resolució és de 0.276 mm per revolució.

A més a més, cal tindre en compte que el traçat de la trajectòria que aquesta tècnica ofereix és relativa al moment d'inici de la prova, fet que afecta greument a l'exactitud del sistema, ja que acumula error des del moment en què comença la prova i, per tant, com més prolongada sigui la trajectòria realitzada, més inexacte serà el resultat.

- Odometria mitjançant llum ultraviolada

Es decideix utilitzar aquesta tècnica per tal de poder obtenir una odometria que es pugui considerar com a precisa. La raó per a la qual es considera una tècnica precisa és que és un sistema que no acumula error amb el temps i que, per tant, cada mesura de les coordenades està referenciada a les coordenades d'inici de la prova i no a les últimes coordenades de posició obtingudes.

En aquesta odometria s'utilitza tecnologia de la companyia *Optitrack*. Aquesta odometria utilitza llum ultraviolada i objectes que la reflecteixen així com càmeres per a localitzar el robot mòbil en un espai preestablert. En la Figura 7.2 s'hi observa una de les 10 càmeres instal·lades en l'entorn de la Figura 7.5. Aquestes capturen la posició de les esferes de color blanc marcades en la Figura 7.3, que són d'un material altament reflectant a la llum ultraviolada.

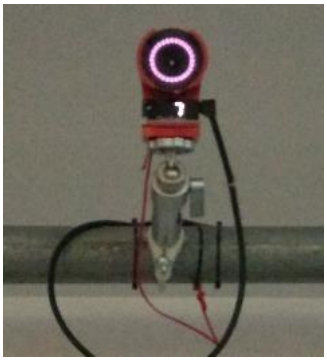

Figura 7.2. Càmera *Flex 13* d'*Optitrack* que inclou els LEDs que emeten llum ultraviolada.

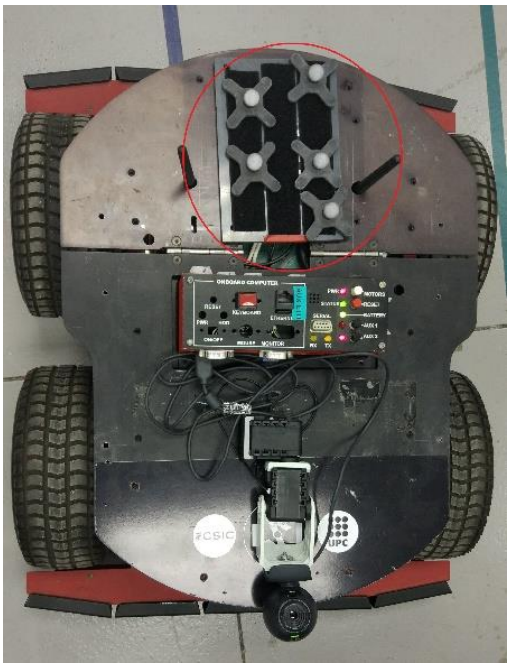

Figura 7.3. Material reflectant a ones de llum ultraviolada emprat per la tecnologia d'*Optitrack.*

Com que la llum ultraviolada és una de les components de la llum solar, és necessari que l'entorn on es realitzen les proves estigui aïllada d'aquesta, ja que això ocasionaria que poguessin aparèixer noves regions no desitjades en el sistema d'*Optitrack*.

En la Figura 7.4 s'hi poden observar els eixos que s'han establert en la càmera que captura les imatges de les proves realitzades.

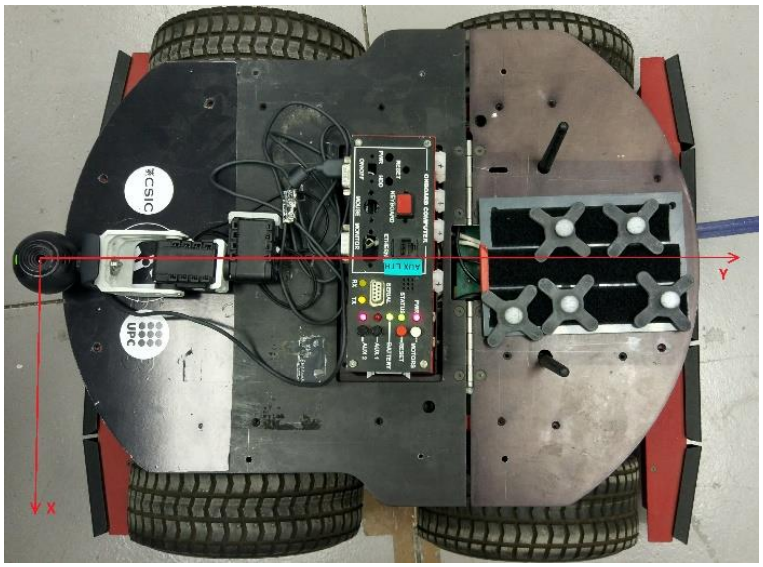

Figura 7.4. Eixos aplicats al robot utilitzat per les proves en un entorn de laboratori.

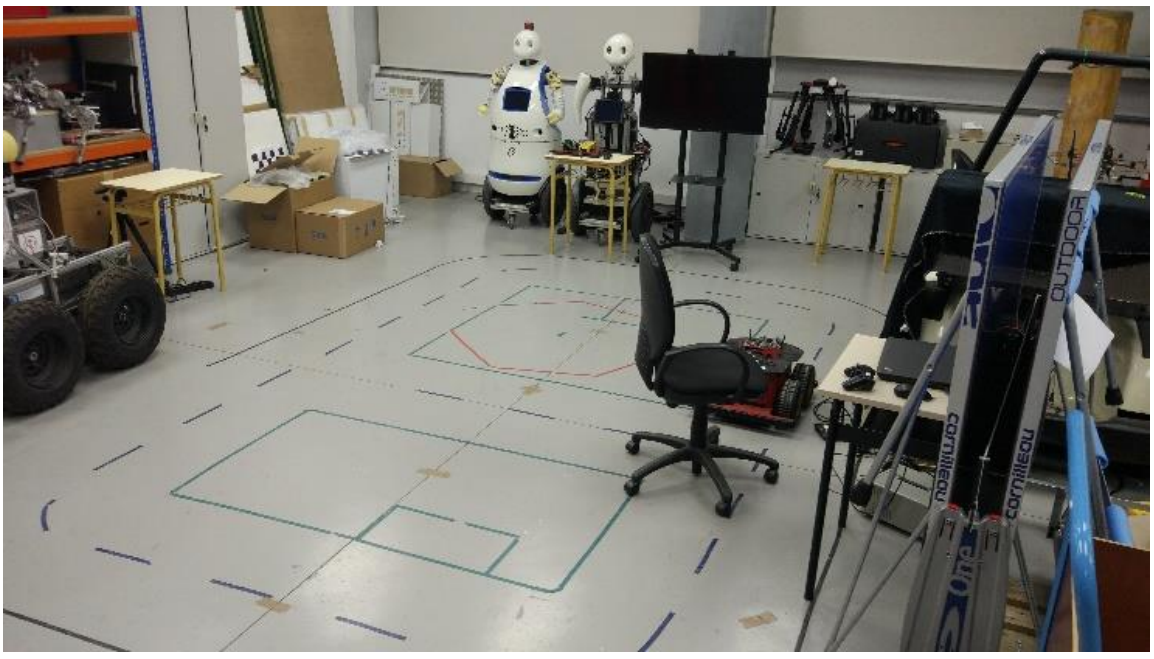

Figura 7.5. Entorn de laboratori on es realitzen les proves de comparació dels sistemes d'odometria.

Pel que fa a la realització dels tests, cal esmentar que s'ha emprat ROS i el programa propi d'*Optitrack* anomenant *Movie Tracker*. Mitjançant aquests es capturen simultàniament les imatges de la càmera, les posicions dels codificadors de les rodes i les posicions capturades pel sistema d'*Optitrack*. És només d'aquesta manera com es pot assegurar que les trajectòries capturades per cada un d'aquests sistemes hagin sigut obtingudes en les mateixes circumstàncies.

Tanmateix, les possibilitats de les proves que poden realitzar-se en aquest entorn són bastant limitades, ja que sigui per l'àrea que cobreix la instal·lació de les càmeres d'*Optitrack* o perquè, amb les possibilitats del sistema desenvolupat, les trajectòries realitzables no són totes les desitjables per la configuració d'il·luminació de l'entorn disponible.

7.1. Primer test per la comparació de les odometries

Com a primer test en l'entorn del laboratori de l'IRI es realitza una prova que pot dividir-se en 3 fases. En la primera, el robot es desplaça en línia recta fins a una posició indicada. En la segona, realitza un canvi de sentit de 180º per a desfer el desplaçament inicial fins a aproximadament el punt d'origen de la prova en la tercera de les fases. Tenint en compte que l'odometria que millor reflecteix la trajectòria real del robot és la del sistema *Optitrack*, s'aprecia en la Figura 7.6 com la trajectòria realitzada no correspon exactament a la forma en "U" que es descriu sinó que després de realitzar el gir de 180º el robot surt amb un cert angle que fa que la trajectòria de tornada no sigui una línia vertical pura.

Per tal de poder fer la comparació de les odometries més efectiva, les tres s'han traçat compartint el punt l'origen de la prova a aproximadament (1.1 , 2.5).

A més a més, i perquè el sistema pugui traçar la trajectòria realitzada de manera correcta, cal modificar els paràmetres referents a l'alçada de la càmera fins al sostre que equivaldrà a 295 centímetres, així com l'angle de visió de la càmera que el robot mòbil utilitza que equival a 60º. A més a més, en aquest cas el llindar de binarització s'estableix a 0,85 i l'amplada del fotograma a 640 píxels.

De la Figura 7.6, tenint en compte que l'odometria obtinguda mitjançant *Optitrack* es considerada com a la traçada de referència, se'n poden extreure vàries conclusions.

Observant a la traçada de la trajectòria realitzada per l'odometria dels codificadors de les rodes del robot mòbil, està clar que mentre aquest realitza el canvi de sentit de 180º les rodes que tenen els codificadors rellisquen, obtenint així la recta esbiaixada de tornada.

S'aprecia també com la mesura del desplaçament al final de la primera fase és consistent en les tres odometries, podent afirmar que l'odometria visual monocular desenvolupada en aquest projecte és la que més s'assimila a la de referència.

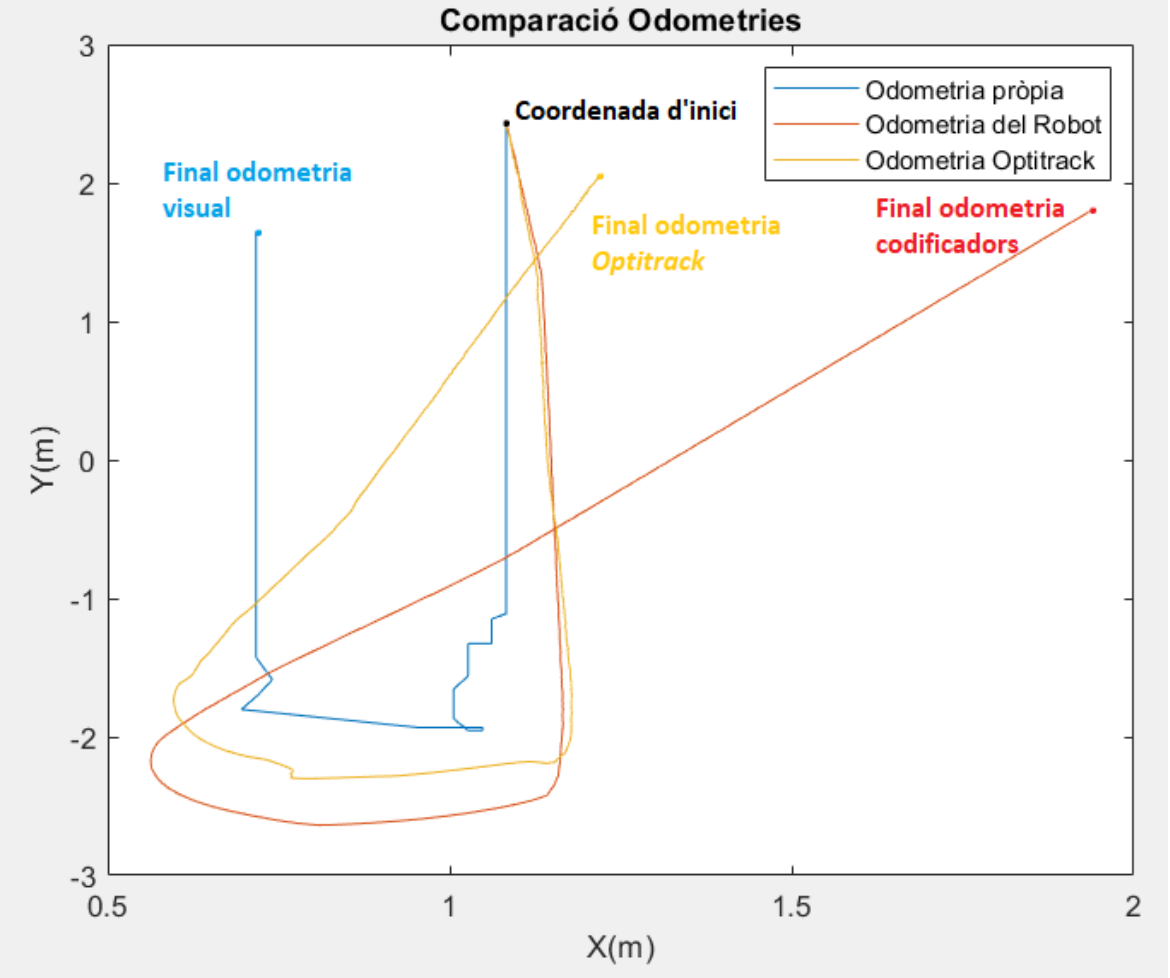

Figura 7.6. Comparació dels sistemes d'odometria en un gir de 180º.

Cal comentar la dificultat del sistema desenvolupat de detectar de manera correcta quin és l'angle de sortida de la fase 2, ja que tal com pot apreciar-se en la traçada de l'*Optitrack*, la traçada real no correspon a una línia purament vertical, tal com s'intueix del resultat de l'odometria realitzada en aquest projecte. Això és causa del fet que, tal com es comenta en Abast del treball, els angles que es tracten en aquest projecte són solament els de 90 i 180º.

Per tal de realitzar el càlcul de l'angle, en aquesta prova el sistema té com a entrada simplement una regió activa en el fotograma en el moment del canvi de sentit de 180º. És per això que el lector podrà denotar el moment en què s'han detectat els dos girs de 90º necessaris per a realitzar el canvi de sentit complet. Aquest càlcul és d'especial dificultat, ja que tal com es mostrava en la Figura 6.7, l'aplicació ha de ser capaç de detectar els girs de manera consecutiva.

7.1.1. Càlcul del percentatge d'error absolut

Per tal de poder definir l'exactitud de la tècnica desenvolupada en aquest treball es calcularà el percentatge d'error absolut que hi ha de mitjana en les coordenades de les traçades de la trajectòria capturada per l'odometria establerta pels codificadors de les rodes i l'odometria visual d'aquest treball. S'utilitzarà com a traçada de referència la que s'obté mitjançant el sistema d'*Optitrack,* ja que la seva precisió mil·limètrica assegura fidelitat amb la trajectòria realitzada en la realitat.

Així doncs, per calcular l'error absolut de cada punt primerament es calcularà l'error que tenen les coordenades respecte a les quals s'agafen de referència, tal com es mostra en les equacions 7.1 i 7.2.

$$Error\ en\ un\ punt_x = \frac{|Coordenada_{x_{Optitrack}} - Coordenada_{x_{odometria}}|}{|Coordenada_{x_{Optitrack}}|} \quad \textbf{(Eq. 7.1)}$$

$$Error\ en\ un\ punt_y = \frac{|Coordenada_{y_{Optitrack}} - Coordenada_{y_{odometria}}|}{|Coordenada_{y_{Optitrack}}|} \quad \textbf{(Eq. 7.2)}$$

Amb l'equació 7.3 es calcularà error que hi ha en cada un dels punts de la traçada.

$$Error\ en\ un\ punt = Error\ en\ un\ punt_x + Error\ en\ un\ punt_y \quad \textbf{(Eq. 7.3)}$$

Cada un dels valors d'error en els punts de la traçada de trajectòria se suma i es divideix pel nombre total de punts tal com es mostra en l'equació 7.4.

El càlcul d'aquest error correspon a la mitjana absoluta del percentatge d'error o MAPE (Mean Absolute Percentatge Error)

$$MAPE = \frac{\sum_{t=1}^{n} Error\ en\ un\ punt}{n} \quad \textbf{(Eq. 7.4)}$$

Aquest error mostra la precisió de cada una de les odometries respecte a les que es considera com a ideal.

Taula 7.1. Mitjana del percentatge d'error absolut de les odometries.

	MAPE
ODOMETRIA CODIFICADORS DE LES RODES	113.68%
ODOMETRIA VISUAL DESENVOLUPADA	136.41%

Abans de passar a comentar els resultats obtinguts és necessari esmentar que les proves que van realitzar-se en el laboratori incloïen algunes de les condicions que el sistema visual monocular pot trobar-se. Les raons principals són les següents:

- Pot observar-se en la Figura 7.7 com les llums del sostre que hi ha en el laboratori de l'IRI difícilment ressemblen les presents en habitatge comú. Això provoca que, tal com s'observa en la imatge on s'ha aplicat la transformada de distàncies de la figura que ens ocupa és complicat discernir quin és el píxel de màxima intensitat d'aquesta, causa que provoca que l'algorisme encarregat de seleccionar-lo pugui seleccionar píxels que, entre dos fotogrames consecutius, no corresponguin al mateix punt de l'objecte i per tant desmereixin el càlcul del desplaçament.
- A causa de les dimensions de les llums, és sovint que aquestes toquen les cantonades dels fotogrames i que, per tant gràcies al Fragment de codi 4.3 s'elimina la regió. En aquest moment en el fotograma no hi ha cap regió activa i per tant no és possible determinar el desplaçament del robot.

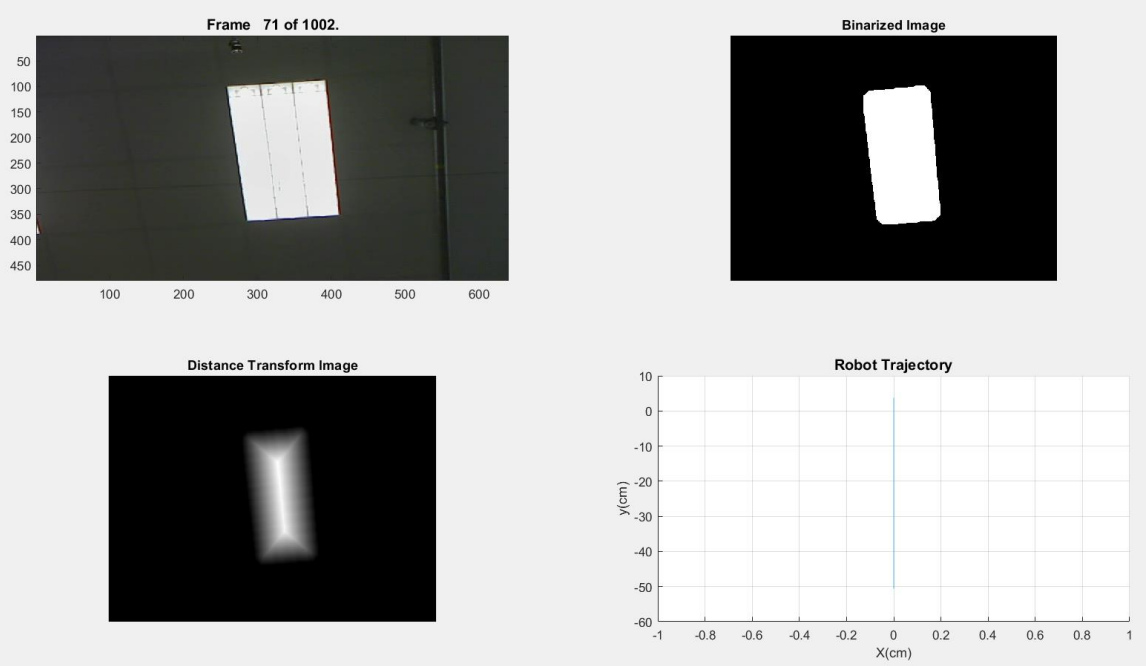

Figura 7.7. *Workspace* del processament del vídeo del laboratori de l'IRI.

Tanmateix, tal com pot extreure's dels resultats de la mitjana del percentatge de l'error absolut, en aquesta prova no s'hi aprecia una gran diferència entre els dos mètodes d'odometria. Dels resultats es pot apreciar que mitjançant les dues tècniques l'error en la traçada de les trajectòries és considerable.

Una de les causes principals és que les dues tècniques acumulen error a mesura que van prenent-se noves mesures, ja que aquestes seran relatives a l'última posició coneguda del robot i no al punt d'inici.

S'aprecia com, en el cas de la tècnica d'odomertia visual desenvolupada en aquest projecte, les coordenades tindran una diferència que de mitjana serà de 136.41% del valor real que haurien de prendre. Tanmateix, aquest error no es considera com quelcom crític que indiqui que el sistema és inviable. Solament indica la diferència al valor que idealment el sistema hauria d'obtenir és per això que dependrà de les condicions i desplaçaments que l'usuari de la tècnica desenvolupada estigui disposat a acceptar.

7.2. Segon test per la comparació de les odometries

Ja com a segon test en l'entorn del laboratori de l'IRI es realitza una trajectòria en la qual es duen a terme quatre girs de 90º, tal com es mostra en l'odometria del sistema *Optitrack* de la Figura 7.8. En aquest cas, el punt inicial respecte el qual s'han traçat les trajectòries desitjades està localitzat aproximadament a les coordenades (1.4, 0.5).

En aquesta prova es mantenen els paràmetres que s'han definit en la prova anterior pel que fa al FOV i a la distància de la càmera al sostre.

Prenent l'odometria traçada pel sistema d'*Optitrack,* s'aprecia en la figura en qüestió com l'odometria que més ressembla la trajectòria de referència és la traçada per l'odometria dels codificadors de les rodes del robot mòbil. Analitzant aquesta en detall s'observa que, en cada un dels girs de 90º que detecta l'odometria guiada pels codificadors, l'error acumulat s'incrementa de manera destacable fins a arribar a diferenciar-se aproximadament 50 centímetres de la trajectòria de referència. En aquest cas no es pot afirmar que la font de major incertesa d'aquesta tècnica sigui la relliscada de les rodes amb els codificadors sinó la mateixa acumulació d'error en el transcurs de la prova.

Analitzant ara els resultats de la tècnica desenvolupada en aquest projecte s'aprecia com en primera instància la traçada no ha sigut tan exacta com es desitjava. En aquest s'observa com la distància de la trajectòria traçada pel sistema d'odometria visual respecte a la trajectòria de referència arriba als 50 centímetres com a màxim. Això es deu al fet que, tal com es mostra en la traçada de la trajectòria de referència, els girs de 90º que realitza el robot no són bruscos sinó que més aviat gira suaument a mesura que avança. Això fa que el càlcul de l'angle relatiu acumulat que realitza el Fragment de codi 6.1 tardi més temps a acumular els 91º mínims necessaris per a traçar aquest canvi de sentit i que, per tant, hi hagi una diferència substancial entre el gir real i el gir percebut pel sistema desenvolupat.

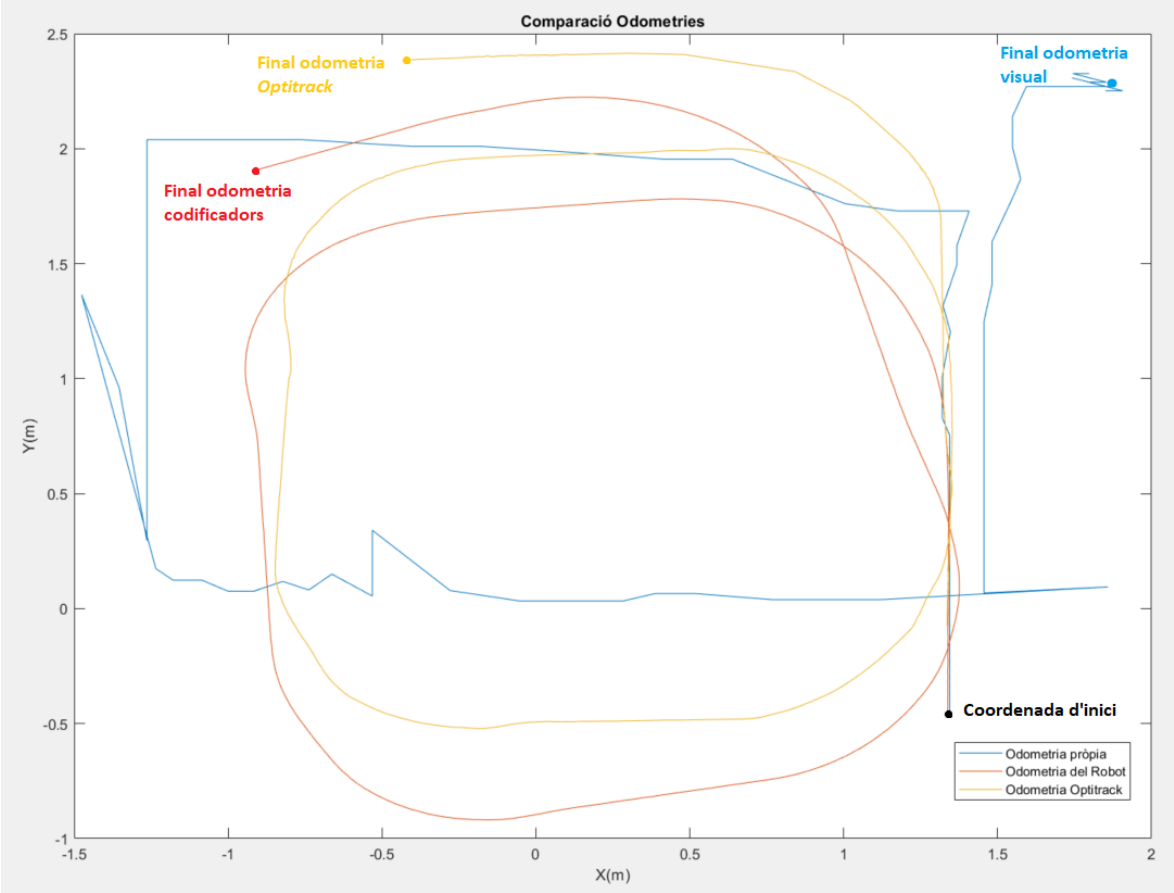

Figura 7.8. Comparació dels sistemes d'odometria realitzant una trajectòria que conté quatre girs de 90º.

Cal destacar també la destacable distància entre les coordenades finals del sistema visual desenvolupat envers les altres dues tècniques d'odometria. Aquesta es deu al fet que en el moment en el qual el sistema hauria de detectar l'últim dels girs de 90º, hi ha dues llums les regions equivalents de les quals toquen amb les cantonades dels fotogrames, fet que impossibilita detectar qualsevol tipus de desplaçament que el robot realitzi.

7.2.1. Càlcul del percentatge d'error absolut

Per tal de calcular el percentatge d'error de cada una de les odometries respecte a l'odometria de referència s'empren les mateixes equacions i procediments que els ja explicats en la secció 7.1.1.

Els resultats obtinguts són els que apareixen en la Taula 7.2.

Taula 7.2. Mitjana del percentatge d'error absolut de les odometries

	MAPE
ODOMETRIA CODIFICADORS DE LES RODES	127.6%
ODOMETRIA VISUAL DESENVOLUPADA	134.37%

En aquesta prova, tal com era previsible després d'analitzar la Figura 7.8, els errors d'ambdós odometries són relativament elevats, a destacar l'error de l'odometria visual desenvolupada en aquest projecte.

Les raons principals per a entendre perquè el sistema desenvolupat no és tan exacte com caldria esperar són compartides amb les explicades en la prova que s'ha tractat en la secció 7.1.

8. Anàlisi de l'impacte ambiental

Aquest projecte no té una repercussió mediambiental especialment significativa agafant com a referència l'impacte que un individu realitza de mitjana en un període de temps comparable. Tanmateix, és obvi que cal tindre en compte l'energia necessària per tal que els sistemes emprats (ordinadors, robots i sistema d'Optitrack) funcionin correctament, tot i que aquesta és bastant reduïda.

El robot emprat en l'habitatge familiar consumeix una mica més de 27 W i el robot Pioneer 3-AT consumeix una mica més de 50 W, sent ambdós consums certs durant els tests. Pel que fa als ordinadors utilitzats, aquests tenen associat el percentatge més gran de consum d'energia. De fet, el consum de l'energia de l'ordinador emprat per a la realització d'aquest projecte acabarà amb la finalització d'aquest. La major problemàtica en el projecte està relacionada amb el silici emprat en els robots i ordinadors del projecte en les seves unitats CPU (*Control Process Unit*) i circuits integrats.

Un altre fet en el qual aquest projecte pot haver afectat el medi ambient és la fabricació del robot que s'han emprat en les proves de la tècnica desenvolupada en aquest treball. Tanmateix, aquest tema per la seva complexitat i perquè els robots no han sigut construïts únicament per aquest projecte, l'anàlisi d'aquest procés no entra en l'abast d'aquest treball.

Per últim, afegir que tot el *software* utilitzat i desenvolupat en aquest projecte no suposa cap degradació mediambiental en si mateix.

Resumint, aquest projecte no implica una gran repercussió mediambiental, ja que al tractar-se d'un treball que se centra en gran mesura en desenvolupament de *software*, el h*ardware* en el qual s'ha provat el sistema desenvolupat no ha sigut en cap cas exclusiu a aquest treball.

Conclusions

El principal objectiu d'aquest treball era el de provar de desenvolupar els principis d'una tècnica d'odometria de cost molt reduït capaç de mimetitzar comportaments de sistemes molt més cars. Aquest objectiu es treballa quan es decideix que el sistema que es desenvoluparà basarà el seu funcionament en maquinari que està àmpliament disponible en el mercat a preus molt econòmics i centrarà les possibilitats del sistema en el desenvolupament d'un *software* relativament complex.

Amb la realització d'aquest treball s'ha pogut comprovar que és possible desenvolupar un sistema de baix cost capaç de traçar trajectòries de robots mòbils en habitatges comuns. Tanmateix, s'han hagut d'assumir algunes situacions en les quals la tècnica basada en odometria visual o bé no funcionarà o s'obtindran resultats insatisfactoris. En ser un sistema que basa el seu funcionament en la detecció de punts d'interès de focus de llums localitzats en el sostre, quan aquests es troben apagats, difícilment distingibles per les condicions lumíniques de l'entorn o simplement inexistents en l'habitació on hi ha el robot, el sistema és incapaç de detectar el desplaçament o angles relatius del robot. A més a més, els canvis de lluminositat bruscs poden suposar que les propietats de les regions detectades canviïn dràsticament i que aleshores el sistema no sigui capaç de detectar els punts d'interès apropiats. La quantitat de possibilitats que el sistema ha de ser capaç de detectar i processar correctament és molt gran i és per això perquè el processament de les imatges ha de ser suficientment flexible com poder adaptar-se a aquests canvis.

Aquest projecte és una primera iteració per al desenvolupament d'un sistema basat en odometria visual monocular capaç de traçar correctament la trajectòria realitzada per un robot mòbil. Com a primera iteració i al basar-se en un sistema basat en visió per computador que s'ha desenvolupat des de zero, és una tècnica que presenta limitacions pel que fa a els entorns en els quals és capaç d'operar correctament. Tanmateix, havent definit els entorns adequats el sistema és capaç de detectar satisfactòriament els desplaçaments i angles relatius així com de traçar de manera abastant precisa quina ha sigut la trajectòria del robot.

MATLAB® es mostra una eina excepcional per a les aplicacions de processament d'imatges. Les llibreries disponibles ofereixen moltes possibilitats al programador i, a més a més, simplifiquen en gran mesura moltes de les tècniques per al processament d'imatges així com la integració del sistema que es desenvolupi amb càmeres externes per a l'aplicació que treballa a temps real.

És destacable que per a la realització d'aquest projecte s'han hagut d'ampliar els coneixements prèviament assolits durant els estudis d'enginyeria electrònica industrial i automàtica en el tractament d'imatges per tal de discernir les operacions de processament de fotogrames adequades per a detectar les regions desitjades amb la menor quantitat d'interferències. Tanmateix, actualment hi ha una

enorme quantitat de tècniques de processament d'imatges, essent les tècniques basades en m*achine learning* l'aparent punta de llança de les tecnologies que irromp en aquest camp. Tot i això, les limitacions en els coneixements de les mateixes així com de la quantitat d'informació disponible i del temps requerit en l'elaboració d'aquest projecte van fer que es decidís que aquestes quedaven fora de l'abast del treball.

Treball Futur

Com a treball basat en visió per computador, hi ha moltes situacions en la vida real que s'han hagut de simplificar per tal de poder desenvolupar una primera iteració fiable.

Una de les simplificacions que hauria de centralitzar propers desenvolupaments en la tècnica d'aquest projecte és la millora en el traçat de trajectòries i el tractament que es dóna als angles que no són ±90º o 180º que el sistema calcula. D'aquesta manera seria possible mostrar trajectòries de manera molt més precisa.

D'altra banda i per tal de millorar el sistema desenvolupat es proposen les següents millores, pendents d'implementar per tal de comprovar si realment suposarien una millor precisió en la traçada de la trajectòria del robot mòbil.

- La primera de les millores possibles és la d'afegir un telèmetre làser des del qual poder obtenir la distància de la càmera al sostre per cada fotograma. D'aquesta manera seria possible que el sistema desenvolupat en aquest projecte pogués funcionar correctament sense haver d'ajustar els paràmetres del sistema.
- La segona de les millores serviria perquè el sistema no tingués errors de mesura en aquelles situacions en les quals les llums del sostre tingui formes que dificultin trobar punts d'intensitat clarament superiors a la resta. Per tal d'implementar aquesta millora una possibilitat seria la d'utilitzar el centroide de la regió i combinar-lo amb el punt de màxima intensitat de la regió on s'ha aplicat la transforma de distàncies i combinar-los.
- La tercera de les possibles millores seria la de fusionar les odometries dels codificadors de les rodes i la desenvolupada en aquest treball. D'aquesta manera seria possible millorar aquelles trajectòries obtingues en situacions en les quals, per exemple, les rodes on hi ha els codificadors rellisquen en la superfície en la qual el robot mòbil es desplaça.

Aquest treball només cobreix una de les àrees del projecte que s'està desenvolupant a l'IRI. És per això que, tan bon punt s'hagi perfeccionat la tècnica que es planteja en aquest projecte els següents passos seran la implementació de tot el sistema en un sol robot mòbil que desenvolupi les funcions designades com a cadira de rodes.

Pressupost i/o Anàlisi Econòmica

En aquest apartat es realitza una anàlisi dels costos associats al desenvolupament d'aquest projecte. L'anàlisi econòmica es realitzarà diferenciant tres conceptes diferents. En primer lloc es valorarà els costos que presenta els dispositius que s'han emprat en la realització d'aquest treball. Seguidament, els costos de test i desenvolupament. Per últim, els recursos humans relacionats amb el desenvolupament de la tècnica visual monocular, les proves tant en l'entorn de laboratori com en l'entorn de l'habitatge familiar i documentació del projecte.

Concepte	Preu unitari (€)	Quantitat (unitats)	Cost (€)	Cost amb l'IRI (€)
Càmera Flex 13	1251	10	12521	0
Hardware Key	99	2	198	0
OptiHub 2	299	6	1794	0
CS-200 Calibration Square	149	2	298	0
CW-500 Calibration Wand Kit	299	2	598	0
12.7mm M4 Thread Markers	50	2	100	0
Rigid Body Marker Base	5	4	20	0
Sync Cable: 10m	10	4	40	0
USB 2.0 Active Extension Cable : 5m	20	6	120	0
USB Cable High Grade, Down Angle: 5m	10	24	240	0
USB Uplink Cable, A to B: 5m	5	6	30	0
Robot Pioneer 3-AT	4000	1	4000	0
Logitech E 3560 QuickCamera	15	1	15	0
iRobot Roomba 880	330	1	330	330
Total			**20304**	**330**

Cal remarcar que el sistema *Optitrack*, que en total suma un cost de 15959 €, és la infraestructura més cara utilitzada en aquest projecte. Si bé aquest treball podria haver-se dut a terme sense aquest sistema d'odometria, en aquest supòsit no s'hauria tingut una odometria de referència per tal de poder avaluar amb el menor marge d'error la tècnica desenvolupada en aquest treball.

A continuació es realitza un desglossament de preus de recursos de test i desenvolupament.

Concepte	Preu (€)	Preu amb l'IRI (€)
ROS Indigo	0	0
Motive: Tracker	999	0
MATLAB® per estudiants	69	0
PC laboratori	1000	0
Portàtil	750	750
Total	**2143**	**750**

Tal com s'aprecia en els preus totals tant dels sistemes com dels recursos de test i desenvolupament aquest treball no hauria estat possible sense l'activa col·laboració de l'Institut de Robòtica i Informàtica Industrial

Finalment, s'analitzen els recursos humans necessaris per a la realització del projecte.

Concepte	Duració (h)	Preu (€/h)	Cost (€)
Desenvolupament	650	30	19500
Proves al laboratori	50	30	1500
Proves al laboratori (tècnic de laboratori)	5	40	200
Proves en habitatge familiar	20	30	600
Redacció del document	120	30	3600
Total			**25400**

Per últim, es mostra el preu total del projecte desglossat en els tres conceptes esmentats.

Concepte	Preu (€)	Preu amb l'IRI (€)
Recursos dels sistemes emprats	20304	330
Recursos de test i desenvolupament	2143	750
Recursos humans	25400	25400
Total	**47847**	**26480**

Bibliografia

[1] Das, P. et al. Measurement of Displacement and Velocity of a Moving Object from Real Time Video. A: *International Journal of Computer Applications*. 2012, Vol. 49, núm. 13, p. 12-16. ISSN 09758887. DOI 10.5120/7685-0992.

[2] Precise localization of a UAV using visual odometry T . H . (Tjark) Post. A: . 2015.

[3] Yousif, K., Bab-Hadiashar, A. i Hoseinnezhad, R. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. A: *Intelligent Industrial Systems* [en línia]. Springer Singapore, 2015, Vol. 1, núm. 4, p. 289-311. ISSN 2363-6912. DOI 10.1007/s40903-015-0032-7.

[4] McGarey, P. Visual Odometry. A: *University of Toronto*. 2016,

[5] Fraundorfer, F. i Scaramuzza, D. Visual odometry: Part II: Matching, robustness, optimization, and applications. A: *IEEE Robotics and Automation Magazine*. 2012, Vol. 19, núm. 2, p. 78-90. ISSN 10709932. DOI 10.1109/MRA.2012.2182810.

[6] Robotics, A. Pioneer 3 operations manual. A: [en línia]. 2006, p. 4-34. Disponible a: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Pioneer+3+Operations+Manual#0.

Annex A

En l'annex s'adjunten els codis emprats per al processament dels fotogrames així com el codi necessari per al càlcul del percentatge d'error absolut i la creació de les gràfiques per a la comparació de les odometries.

A1. Codi pel processament dels fotogrames d'un vídeo

```
2    % Treball de fi de Grau
3    % Títol: Desenvolupament d'un sistema d'exploració per a robots mòbils
4    % Autor: Jaume Alavedra Mas
5    % Convocatoria: Juny 2018
6    % Director: Sebastián Tornil Sin
7
8    clc;    % Clear the command window.
9    close all;  % Close all figures (except those of imtool.)
10   imtool close all;  % Close all imtool figures.
11   clear;  % Erase all existing variables.
12   workspace;  % Make sure the workspace panel is showing.
13   fontSize = 12;
14   pos_x_image_update = 0;
15   pos_y_image_update = 0;
16
17   % First get the folder that it lives in.
18   folder = fileparts(which('test_180_laboratori.avi')); % Determine where folder is.
movieFullFileName = fullfile(folder, 'test_180_laboratori.avi');
19   % Check to see that it exists.
20   if ~exist(movieFullFileName, 'file')
21       strErrorMessage = sprintf('File not found:\n%s\nYou can choose a new one, or
     cancel', movieFullFileName);
22       response = questdlg(strErrorMessage, 'File not found', 'OK - choose a new movie.',
     'Cancel', 'OK - choose a new movie.');
23       if strcmpi(response, 'OK - choose a new movie.')
24           [baseFileName, folderName, FilterIndex] = uigetfile('*.avi');
25           if ~isequal(baseFileName, 0)
26               movieFullFileName = fullfile(folderName, baseFileName);
27           else
28               return;
29           end
30       else
31           return;
32       end
33   end
34
35   try
36       videoObject = VideoReader(movieFullFileName);
37       % Determine how many frames there are.
38       numberOfFrames = videoObject.NumberOfFrames;
39       vidHeight = videoObject.Height;
40       vidWidth = videoObject.Width;
41
42       numberOfFramesWritten = 0;
43       % Prepare a figure to show the images in the upper half of the screen.
44       figure;
45       set(gcf, 'units','normalized','outerposition',[0 0 1 1]);
46
47       % Ask user if they want to write the individual frames out to disk.
48       promptMessage = sprintf('Do you want to save the individual frames out to
     individual disk files?');
49       button = questdlg(promptMessage, 'Save individual frames?', 'Yes', 'No', 'Yes');
50       if strcmp(button, 'Yes')
51           writeToDisk = true;
52
53           % Extract out the various parts of the filename.
54           [~, baseFileName, extentions] = fileparts(movieFullFileName);
```

```
55          % Make up a special new output subfolder for all the separate
56          % movie frames that we're going to extract and save to disk.
57          folder = pwd;   % Make it a subfolder of the folder where this m-file lives.
58          outputFolder = sprintf('%s/Movie Frames from %s', folder, baseFileName);
59          % Create the folder if it doesn't exist already.
60          if ~exist(outputFolder, 'dir')
61              mkdir(outputFolder);
62          end
63      else
64          writeToDisk = false;
65      end
66
67      % Loop through the movie, writing all frames out.
68      % Each frame will be in a separate file with unique name.
69      meanGrayLevels = zeros(numberOfFrames, 1);
70      meanRedLevels = zeros(numberOfFrames, 1);
71      meanGreenLevels = zeros(numberOfFrames, 1);
72      meanBlueLevels = zeros(numberOfFrames, 1);
73      for frame = 1:5: numberOfFrames
74          % Extract the frame from the movie structure.
75          thisFrame = read(videoObject, frame);
76
77          % Display it
78          hImage = subplot(2, 2, 1);
79          image(thisFrame);
80          caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
81          title(caption, 'FontSize', fontSize);
82          drawnow; % Force it to refresh the window.
83
84          if frame == 1
85              threshold = 0.85; %0.95
86              pos_x_image_gran_update = 0;
87              pos_y_image_gran_update = 0;
88              pos_x_image_petita_update = 0;
89              pos_y_image_petita_update = 0;
90              pos_x_image_res_gran = 0;
91              pos_y_image_res_gran = 0;
92              pos_x_image_res_petita = 0;
93              pos_y_image_res_petita = 0;
94              pos_x_image_gran = 0;
95              pos_y_image_gran = 0;
96              pos_x_image_petita = 0;
97              pos_y_image_petita = 0;
98              inte_gran = 0;
99              inte_petita = 0;
100             exep1 = 0;
101             exep2 = 0;
102             one_blob = 0;
103             mini_counter = 0;
104             apear = 0;
105             cont_disapear = 0;
106             cont_apear=0;
107             disapear = 0;
108             cont_first_blob = 0;
109             first_blob = 0;
110             area_update = 0;
111             camera_span = 52;
112             distance2ceiling =295;
113             h = 2 * distance2ceiling * sin(camera_span / 2);
114             b = distance2ceiling * cos(camera_span / 2);
115             frame_width = 640;
116             frame_higth = 480;
117             image2real_x = b/frame_width;
118             image2real_y = h/frame_higth;
119             pos_x_real = 0;
120             pos_y_real = 0;
121             angle_res180 = 0;
122             accu_pos_x_real = 0;
123             accu_pos_y_real = 0;
124             ii = 0;
125             jj = 0;
126             angle_cont = 0;
```

```
127             accu_angle_res = 0;
128             accu_angle_res_update = 0;
129             angle_exception_right = 0;
130             angle_exception_left = 0;
131             pos_y_real_reverse = 0;
132             angle180_exception = 0;
133             angle_update180 = 0;
134             accu_angle180_res = 0;
135             accu_angle180_res_update = 0;
136             angle_axis = 0;
137             angle180_update = 0;
138         end
139
140         % Update user with the progress.  Display in the command window.
141         if writeToDisk
142             progressIndication = sprintf('Wrote frame %4d of %d.', frame,
    numberOfFrames);
143         else
144             progressIndication = sprintf('>>>>>>>Processed frame %4d of %d.', frame,
    numberOfFrames);
145         end
146         disp(progressIndication);
147         numberOfFramesWritten = numberOfFramesWritten + 1;
148         grayImage = rgb2gray(thisFrame); % Convert to gray scale
149         grayImage_eq = histeq(grayImage);
150         s1 = strel('square',10);
151         grayImag_o8 = imopen(grayImage_eq,s1);
152         thresholdLevel = threshold * 255;
153         binaryImage = (grayImag_o8 >thresholdLevel);
154         s2 = strel('disk',20);
155         binaryImage_o = imopen(binaryImage, s2);
156         hPlot = subplot(2, 2, 2);
157         imshow(binaryImage_o, {}),title('Binarized Image'); % Plot the binary image.
158         %-----------------Labeling------------------
159         [L,n] = bwlabel(binaryImage_o);
160         % Regions in contact with the edges of the image
161         a = zeros(1,n);
162         for t=1:n
163             for i=1:size(thisFrame,1)
164                 for j=1:size(thisFrame,2)
165                     if (L((j-1)*size(thisFrame,1)+i)) == t
166                         if
    ((i==1)||(j==1)||(i==size(thisFrame,1))||(j==size(thisFrame,2))&&(binaryImage_o(i,j)==1
    ))
167                             a(t) = 1;
168                         end
169                     end
170                 end
171             end
172         end
173         %-------------------------------------------
174         % Elimination of the regions in contact with the edges of the image
175         for t=1:n
176             for i=1:size(thisFrame,1)
177                 for j=1:size(thisFrame,2)
178                     if (L((j-1)*size(thisFrame,1)+i)) == t
179                         if (a(t) == 1)
180                             binaryImage_o(i,j) = 0;
181                         end
182                     end
183                 end
184             end
185         end
186         %-----------------Labeling------------------
187         [L,n]=bwlabel(binaryImage_o);
188         %----Distance transform----
189         binaryImage_DT = bwdist(~binaryImage_o);
190         % Display the distance transform image
191         subplot(2, 2, 3);
192         imshow(binaryImage_DT, {}),title('Distance Transform Image');
193         props = regionprops(binaryImage_o, 'area');
194         hold on;
```

```
195         area_gran = 0;
196         area_petita = 0;
197         label_gran = 0;
198         label_petita = 0;
199         if ((length(props) >= 1)&&(cont_first_blob == 0))
200             first_blob = 1;
201         end
202         for k = 1 : length(props)
203             area = props(k).Area;
204             if area > area_gran
205                 area_petita = area_gran;
206                 label_petita = label_gran;
207                 area_gran = area;
208                 label_gran = k;
209             elseif ((area > area_petita)&&(area < area_gran))
210                 area_petita = area;
211                 label_petita = k;
212             end
213             if (length(props) == 1)
214                 area_petita = 0;
215             end
216         end
217         labels = [label_gran, label_petita];
218         positions_imatge = [pos_x_image_gran, pos_y_image_gran, pos_x_image_petita,
    pos_y_image_petita];
219         positions_imatge_update = [pos_x_image_gran_update, pos_y_image_gran_update,
    pos_x_image_petita_update, pos_y_image_petita_update];
220         intensities = [inte_gran, inte_petita];
221         pos_x_image_res = [pos_x_image_res_gran,pos_y_image_res_gran,
    pos_x_image_res_petita, pos_y_image_res_petita];
222         for t=1:length(labels)
223             for i=1:size(thisFrame,1)
224                 for j=1:size(thisFrame,2)
225
226                     if (L((j-1) * size(thisFrame,1) + i)) == labels(t)
227                         current_inte = binaryImage_DT(i,j);
228
229                         if (intensities(t) < current_inte)
230                             intensities(t) = current_inte;
231                             positions_imatge(2 * t - 1) = j;
232                             positions_imatge(2 * t) = i;
233                         end
234                     end
235                 end
236             end
237
238             if ((area_petita ~= 0)&&(t == length(labels)))
239                 mini_counter = mini_counter + 1;
240                 u= [(positions_imatge(1)-positions_imatge(3)) (positions_imatge(2)-
    positions_imatge(4)) 0];
241                 v= [0 1 0];
242                 angle180 =  atan2d(norm(cross(u,v)),dot(u,v));
243                 if (mini_counter >= 4)
244                     angle_res180 = (angle180 - angle_update180);
245                 end
246                 angle_update180 = angle180;
247             elseif (area_petita == 0)
248                 mini_counter = 0;
249                 angle_update180 = 0;
250             end
251             if frame > 2
252                 if (((positions_imatge_update(2 * t - 1) ~=
    0)||(positions_imatge_update(2 * t) ~= 0))&&(positions_imatge(2 * t -
    1)==0)&&(positions_imatge(2 * t) == 0))
253                     if ((((area_update-(area_gran * 0.2)) <= area_gran)&&(area_gran <=
    area_update + (area_gran * 0.2)))&&(disapear == 0))
254                         disp('--------------->>Blob-Disappearence_Case_1<<-------------
    -')
255                         one_blob=1;
256                     else
257                         positions_imatge_update(3) = positions_imatge_update(1);
258                         positions_imatge_update(4) = positions_imatge_update(2);
```

```
259                         positions_imatge_update(1) = pos_x_image_petita_update;
260                         positions_imatge_update(2) = pos_y_image_petita_update;
261                         disp('--------------->>Blob-Disappearence_Case_2<<-------------
    -')
262                         one_blob= 1;
263                         apear = 1;
264                         cont_apear = 0;
265                     end
266                 else
267                     cont_disapear= cont_disapear + 1;
268                     if (cont_disapear >= 4)
269                         disapear = 0;
270                         cont_disapear = 0;
271                     end
272                 end
273                 if (((positions_imatge(2*t - 1) ~= 0)||(positions_imatge(2*t) ~=
    0))&&(positions_imatge_update(2*t - 1) == 0)&&(positions_imatge_update(2*t) == 0))
274                     if (((area_update-(area_update*0.2)) <= area_gran)&&(area_gran <=
    area_update + (area_update * 0.2))&&(apear == 0))
275                         disp('--------------->>Blob-Appearence_Case_1<<--------------')
276                         one_blob = 1;
277
278                     else
279                         positions_imatge_update(3) = positions_imatge_update(1);
280                         positions_imatge_update(4) = positions_imatge_update(2);
281                         positions_imatge_update(1) = pos_x_image_gran_update;
282                         positions_imatge_update(2) = pos_y_image_gran_update;
283                         disp('--------------->>Blob-Appearence_Case_2<<--------------')
284                         exep2 = 1;
285                         one_blob = 1;
286                         disapear = 1;
287                         cont_disapear = 0;
288                     end
289                 else
290                     cont_apear = cont_apear + 1;
291                     if (cont_apear >= 4)
292                         apear = 0;
293                         cont_apear = 0;
294                     end
295                 end
296                 if (t == length(labels))
297                     area_update = area_gran;
298                 end
299                 if ((t == 1)||(one_blob == 1)) %The distance in pixels is found
300                     pos_x_image_res_gran  = positions_imatge(1) -
    positions_imatge_update(1);
301                     pos_y_image_res_gran = positions_imatge(2) -
    positions_imatge_update(2);
302                 end
303                 if (t == 2)
304                     pos_x_image_res_petita  = positions_imatge(3) -
    positions_imatge_update(3);
305                     pos_y_image_res_petita = positions_imatge(4) -
    positions_imatge_update(4);
306                     exep1 = 0;
307                 end
308
309                 if (t == 2) % checks the amount of blobs and finds out the actual
    distance
310                     if ((first_blob == 1)&&(cont_first_blob == 0))
311                         disp('First blob with x=0cm and y=0cm')
312                         cont_first_blob = 1;
313                     elseif ((((positions_imatge(2*t - 1) == 0)&&(positions_imatge(2*t)
    == 0))||one_blob == 1)&&(first_blob) == 1)
314                         if (exep2 == 1)
315                             pos_x_real = pos_x_image_res_petita * image2real_x;
316                             pos_y_real = pos_y_image_res_petita * image2real_y;
317                             exep2 = 0;
318                             one_blob = 0;
319                             angle_res180 = 0;
320                         else
321                             pos_x_real = pos_x_image_res_gran * image2real_x;
```

```
322                             pos_y_real = pos_y_image_res_gran * image2real_y;
323                             one_blob = 0;
324                             angle_res180 = 0;
325                         end
326                     elseif ((positions_imatge(2*t - 1) ~= 0)&&(positions_imatge(2*t) ~=
    0)&&(first_blob == 1))
327                         if (((-5 <= pos_x_image_res_gran)&&(pos_x_image_res_gran <=
    5))||((-5 <= pos_x_image_res_petita)&&(pos_x_image_res_petita <= 5)))
328                             pos_x_real = 0;
329                         else
330                             pos_x_real = ((pos_x_image_res_gran +
    pos_x_image_res_petita)/2) * image2real_x;
331                         end
332                         if (((-5 <= pos_y_image_res_gran)&&(pos_y_image_res_gran <=
    5))||((-5 <= pos_y_image_res_petita)&&(pos_y_image_res_petita <= 5)))
333                             pos_y_real = 0;
334                         else
335                            pos_y_real = ((pos_y_image_res_gran +
    pos_y_image_res_petita)/2) * image2real_y;
336                         end
337                     end
338                     if ((abs(pos_x_real) >= (size(thisFrame,1) * image2real_x *
    0.25))||(abs(pos_y_real) >= (size(thisFrame,2) * image2real_y *
    0.25))||(accu_angle180_res >= 80))
339                         disp('Unprecise distance')
340                     else
341                         fprintf('RAW data x= %fcm and y= %fcm\n', pos_x_real,
    pos_y_real)
342                         if ((-2 <= pos_x_real)&&(pos_x_real <= 2))
343                             pos_x_real = 0;
344                         end
345                         if ((-2 <= pos_y_real)&&(pos_y_real <= 2 ))
346                             pos_y_real = 0;
347                         end
348                         fprintf('>Between two blobs the robot moved
    %fdegrees\n',accu_angle180_res)
349                         if (((accu_angle180_res >= 170)&&(pos_y_real_reverse >=
    8))||(angle180_exception == 1))
350                             disp('---->CHANGE 180°<----');
351                             pos_y_real = -pos_y_real;
352                             if ((angle180_exception == 0) && (pos_y_real_reverse >= 8))
353                                 angle180_exception = 1;
354                             elseif ((angle180_exception == 1) && (pos_y_real_reverse >=
    8))
355                                 angle180_exception = 0;
356                            end
357                            pos_y_real_reverse = 0;
358                         end
359
360                         if ((angle_exception_right == 1)||(angle_exception_left == 3))
361                             disp('---->CHANGE_RIGHT<----');
362                             pos_y_real_mom= pos_y_real;
363                             pos_y_real = pos_x_real;
364                             pos_x_real = pos_y_real_mom;
365                             angle_axis = 1;
366                             angle_cont = angle_cont + 1;
367                         elseif ((angle_exception_right == 2)||(angle_exception_left ==
    2))
368                             disp('---->CHANGE_180<----');
369                             pos_y_real = -pos_y_real;
370                             angle_axis = 0;
371                         elseif ((angle_exception_right == 3)||(angle_exception_left ==
    1))
372                             disp('---->CHANGE_LEFT<----');
373                             pos_y_real_mom = pos_y_real;
374                             pos_y_real = -pos_x_real;
375                             pos_x_real = -pos_y_real_mom;
376                             angle_axis = 1;
377                         end
378
379                         u= [(pos_x_real) (pos_y_real) 0];
380                         if (angle_axis == 1)
```

```
381                             if (pos_x_real >= 0)
382                                 v= [1 0 0];
383                             elseif (pos_x_real < 0)
384                                 v= [-1 0 0];
385                             end
386                         elseif (angle_axis == 0)
387                             if (pos_y_real >= 0)
388                                 v= [0 1 0];
389                             elseif (pos_y_real <0)
390                                 v= [0 -1 0];
391                             end
392                         end
393                         angle=  atan2d(norm(cross(u,v)),dot(u,v));
394                         if (angle_axis == 1)
395                             if (pos_y_real < 0)
396                                 angle = -angle;
397                             end
398                         elseif (angle_axis == 0)
399                             if (pos_x_real < 0)
400                                 angle = -angle;
401                             end
402                         end
403                         fprintf('Computed %fdegrees\n',angle)
404                         fprintf('The robot moved x= %fcm and y= %fcm\n', pos_x_real,
    pos_y_real)
405                         accu_pos_x_real=accu_pos_x_real + pos_x_real;
406                         accu_pos_y_real=accu_pos_y_real + pos_y_real;
407
408                         accu_angle_res = accu_angle_res+angle;
409                         accu_angle180_res = accu_angle180_res+angle_res180;
410                         if (accu_angle180_res == accu_angle180_res_update)
411                             accu_angle180_res=0;
412                         else
413                             pos_y_real_reverse = pos_y_real_reverse +1;
414                         end
415                         if (accu_angle_res == accu_angle_res_update)
416                             accu_angle_res=0;
417                         end
418                         accu_angle_res_update = accu_angle_res;
419                         accu_angle180_res_update = accu_angle180_res;
420
421                         if ((accu_angle_res >= 91)&&(angle >= 4))%91
422                             if (angle_exception_left == 0)
423                                 if (angle_exception_right < 4)
424                                     angle_exception_right = angle_exception_right+1
425                                 end
426                             else
427                                 angle_exception_left = angle_exception_left-1
428                             end
429                         elseif ((accu_angle_res <= -91)&&(angle <= -4))
430                             if (angle_exception_right == 0)
431                                 if (angle_exception_left < 4)
432                                     angle_exception_left = angle_exception_left+1
433                                 end
434                             else
435                                 angle_exception_right = angle_exception_right-1
436                             end
437                         end
438
439                         % Plot the trajectory
440                         fprintf('Accumulated of x= %fcm and y= %fcm and %fdegree\n',
    accu_pos_x_real, accu_pos_y_real, accu_angle_res)
441                         ii = [ii accu_pos_x_real];
442                         jj = [jj accu_pos_y_real];
443                         hPlot = subplot(2, 2, 4);
444                         hold on
445                         plot(ii(:),jj(:));
446                         %xlim([-500 100])
447                         %ylim([-400 50])
448                         xy = [ii(:), jj(:)];
449                         %dlmwrite('Test_180_algo.txt', xy, 'delimiter', ',');
450                         title('Robot Trajectory', 'FontSize', fontSize);
```

```
451                         grid on;
452                         xlabel('X(cm)')
453                         ylabel('y(cm)')
454                     end
455                 end
456 
457             end
458             if ((2*t - 1) == 1)
459                 pos_x_image_gran_update  = positions_imatge(2*t - 1);
460             elseif ((2*t - 1) == 3)
461                 pos_x_image_petita_update  = positions_imatge(2*t - 1);
462             end
463             if ((2*t) == 2)
464                 pos_y_image_gran_update = positions_imatge(2*t);
465             elseif ((2*t) == 4)
466                 pos_y_image_petita_update = positions_imatge(2*t);
467             end
468         end
469 
470         % Write the image array to the output file, if requested.
471         if writeToDisk
472             % Construct an output image file name.
473             outputBaseFileName = sprintf('Frame %4.4d.png', frame);
474             outputFullFileName = fullfile(outputFolder, outputBaseFileName);
475             % Stamp the name and frame number onto the image.
476             text(5, 15, outputBaseFileName, 'FontSize', 20);
477             imwrite(binaryImage_DT, outputFullFileName);
478         end
479     end
480 
481     % Alert user that we're done.
482     if writeToDisk
483         finishedMessage = sprintf('Done!  It wrote %d frames to folder\n"%s"',
    numberOfFramesWritten, outputFolder);
484     else
485         finishedMessage = sprintf('Done!  It processed %d frames of\n"%s"',
    numberOfFramesWritten, movieFullFileName);
486     end
487     disp(finishedMessage); % Write to command window.
488     uiwait(msgbox(finishedMessage)); % Also pop up a message box.
489     return
490 
491 catch ME
492     % Some error happened if you get here.
493     strErrorMessage = sprintf('Error extracting movie frames from:\n\n%s\n\nError:
    %s\n\n)', movieFullFileName, ME.message);
494     uiwait(msgbox(strErrorMessage));
495 end
```

A2. Codi pel càlcul del percentatge d'error i per la comparació de les odometries

```
511 % Treball de fi de Grau
512 % Títol: Desenvolupament d'un sistema d'exploració per a robots mòbils
513 % Autor: Jaume Alavedra Mas
514 % Convocatoria: Juny 2018
515 % Director: Sebastián Tornil Sin
516 
517 close all
518 clc
519 %-------------------------
520 %   assign data to variables
521 % ------------------------
522 x = odomrobot{:,6}; % encoders data
523 y = odomrobot{:,7};
```

```
524 x1 = odomoptitrack{:,6}; % Optitrack data
525 y1 = odomoptitrack{:,7};
526 x2 = Test180algo{:,1}; % Developed visual odometry data
527 y2 = Test180algo{:,2};
528 ass_2up = 0;
529 counter = 0;
530 total_error = 0;
531 u = [0 0 1];
532 ii = 0;
533 jj = 0;
534 counter = 0;
535
536 % -------------------------
537 % Code to make the origin of all 3 graphs coincide
538 % -------------------------
539 for i= 1:length(x)
540     n = x(i) - odomrobot{1,6}; % Depending on the data some further constants might
    need to be added
541     x(i) = n + odomoptitrack{1,6};
542 end
543 for j= 1:length(y)
544     m = y(j) - odomrobot{1,7};
545     y(j) = m + odomoptitrack{1,7};
546 end
547 for i= 1:length(x2)
548     n = x2(i) - Test180algo{1,1}; % Depending on the data some further constants might
    need to be added
549     x2(i) = n + odomoptitrack{1,6};
550 end%
551 for j= 1:length(y2)
552     m = y2(j) - Test180algo{1,2};
553     y2(j) = m + odomoptitrack{1,7};
554 end
555
556 % ----------------------
557 % Graph the trajectory of the three odometries
558 % ----------------------
559 odom_prop = plot(x2,y2);
560 hold on
561 robot = plot(x, y);
562 dir_x = (odomrobot{1,6}-odomrobot{300,6});
563 dir_y = (odomrobot{1,7}-odomrobot{300,7});
564 dir = rad2deg(atan2(dir_y,dir_x));
565 real_dir = 180 - dir ;
566 hold on
567 odom_optitrack = plot(odomoptitrack{:,6},odomoptitrack{:,7});
568 legend('Odometria pròpia','Odometria del Robot','Odometria Optitrack')
569 hold off
570 xlabel('X(m)'), ylabel('Y(m)')
571 title('Comparació Odometries')
572 rotate(robot,u, real_dir)
573
574 h = findobj(gca,'Type','line')
575 new_x=get(h(3),'Xdata');
576 new_y=get(h(3),'Ydata');
577 new_x1 = x1;
578 new_y1 = y1;
579 new_x1_resampled = resample(new_x1, length(new_x), length(new_x1));
580 new_y1_resampled = resample(new_x1, length(new_x), length(new_x1));
581
582
583 % -------------
584 % Computation of the mean absolute error
585 % -------------
586 for i = 1:length(new_x1_resampled)
587     error_x = abs(new_x1_resampled(i) - new_x(i)) / abs(new_x1_resampled(i))
588     error_y = abs(new_y1_resampled(i) - new_y(i)) / abs(new_y1_resampled(i));
589     sumat_error = error_x + error_y;
590 end
591 average_error = total_error / counter
```

A3. Codi pel processament dels fotogrames obtinguts a temps real

```
% Treball de fi de Grau
% Títol: Desenvolupament d'un sistema d'exploració per a robots mòbils
% Autor: Jaume Alavedra Mas
% Convocatoria: Juny 2018
% Director: Sebastián Tornil Sin

clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.
fontSize = 12;
pos_x_image_update = 0;
pos_y_image_update = 0;
counter = 0;

try
    cam = webcam('Logitech Webcam 300')
    while(1)
        counter = counter + 1;
        ceiling = snapshot(cam);
        % Display it
        hImage = subplot(2, 2, 1);
        image(ceiling);
        caption = sprintf('Frame %d.', counter);
        title(caption, 'FontSize', fontSize);
        drawnow; % Force it to refresh the window.
        grey_ceiling = rgb2gray(ceiling); %converció a greyscale

        if counter == 1
            threshold = 0.95;
            pos_x_image_gran_update = 0;
            pos_y_image_gran_update = 0;
            pos_x_image_petita_update = 0;
            pos_y_image_petita_update = 0;
            pos_x_image_res_gran = 0;
            pos_y_image_res_gran = 0;
            pos_x_image_res_petita = 0;
            pos_y_image_res_petita = 0;
            pos_x_image_gran = 0;
            pos_y_image_gran = 0;
            pos_x_image_petita = 0;
            pos_y_image_petita = 0;
            inte_gran = 0;
            inte_petita = 0;
            exep1 = 0;
            exep2 = 0;
            one_blob = 0;
            mini_counter = 0;
            apear = 0;
            cont_disapear = 0;
            cont_apear=0;
            disapear = 0;
            cont_first_blob = 0;
            first_blob = 0;
            area_update = 0;
            camera_span = 52;
            distance2ceiling = 240;
            h = 2 * distance2ceiling * sin(camera_span / 2);
            b = distance2ceiling * cos(camera_span / 2);
            frame_width = 720; %640
            frame_higth = 480;
            image2real_x = b/frame_width;
            image2real_y = h/frame_higth;
            pos_x_real = 0;
            pos_y_real = 0;
            angle_res180 = 0;
            accu_pos_x_real = 0;
            accu_pos_y_real = 0;
```

```
70              ii = 0;
71              jj = 0;
72              angle_cont = 0;
73              accu_angle_res = 0;
74              accu_angle_res_update = 0;
75              angle_exception_right = 0;
76              angle_exception_left = 0;
77              pos_y_real_reverse = 0;
78              angle180_exception = 0;
79              angle_update180 = 0;
80              accu_angle180_res = 0;
81              accu_angle180_res_update = 0;
82              angle_axis = 0;
83              angle180_update = 0;
84          end
85
86          fprintf('>Processed frame %d.\n', counter);
87          grayImage_eq=histeq(grey_ceiling);
88          s1 = strel('square',10);
89          grayImag_o8 = imopen(grayImage_eq,s1);
90          thresholdLevel = threshold * 255;
91          binaryImage = (grayImag_o8 >thresholdLevel);
92          s2 = strel('disk',20);
93          binaryImage_o = imopen(binaryImage, s2);
94          hPlot = subplot(2, 2, 2);
95          imshow(binaryImage_o, {}),title('Binarized Image');
96          %-----------------Labeling-----------------
97          [L,n]=bwlabel(binaryImage_o);
98          % Regions in contact with the edges of the image
99          a = zeros(1,n);
100         for t=1:n
101             for i=1:size(ceiling,1)
102                 for j=1:size(ceiling,2)
103
104                     if (L((j-1)*size(ceiling,1)+i))==t
105
106                         if
    ((i==1)||(j==1)||(i==size(ceiling,1))||(j==size(ceiling,2))&&(binaryImage_o(i,j)==1))
107
108                             a(t)=1;
109
110                         end
111                     end
112                 end
113             end
114         end
115
116         %-----------------------------------------------
117         % Elimination of the regions in contact with the edges of the image
118         for t=1:n
119             for i=1:size(ceiling,1)
120                 for j=1:size(ceiling,2)
121
122                     if (L((j-1)*size(ceiling,1)+i))==t
123
124                         if (a(t)==1)
125
126                             binaryImage_o(i,j)=0;
127
128                         end
129                     end
130                 end
131             end
132         end
133         %-----------------Labeling-----------------
134         [L,n]=bwlabel(binaryImage_o);
135         %----Distance transform----
136         binaryImage_DT = bwdist(~binaryImage_o);
137         % Display the distance transform image
138         subplot(2, 2, 3);
139         imshow(binaryImage_DT, {}),title('Distance Transform Image');
140         props = regionprops(binaryImage_o, 'area');
```

```
141         hold on;
142         area_gran = 0;
143         area_petita = 0;
144         label_gran = 0;
145         label_petita = 0;
146         if ((length(props) >= 1)&&(cont_first_blob == 0))
147             first_blob = 1;
148         end
149         for k = 1 : length(props)
150             area = props(k).Area;
151             if area > area_gran
152                 area_petita=area_gran;
153                 label_petita = label_gran;
154                 area_gran = area;
155                 label_gran = k;
156             elseif ((area > area_petita)&&(area< area_gran))
157                 area_petita = area;
158                 label_petita = k;
159             end
160             if (length(props)==1)
161                 area_petita=0;
162             end
163         end
164         labels = [label_gran, label_petita];
165         positions_imatge = [pos_x_image_gran, pos_y_image_gran, pos_x_image_petita,
    pos_y_image_petita];
166         positions_imatge_update = [pos_x_image_gran_update, pos_y_image_gran_update,
    pos_x_image_petita_update, pos_y_image_petita_update];
167         intensities = [inte_gran, inte_petita];
168         pos_x_image_res = [pos_x_image_res_gran,pos_y_image_res_gran,
    pos_x_image_res_petita, pos_y_image_res_petita];
169         for t=1:length(labels)
170             for i=1:size(ceiling,1)
171                 for j=1:size(ceiling,2)
172
173                     if (L((j-1)*size(ceiling,1)+i))==labels(t)
174                         current_inte = binaryImage_DT(i,j);
175
176                         if (intensities(t)<current_inte)
177                             intensities(t) = current_inte;
178                             positions_imatge(2*t - 1) = j;
179                             positions_imatge(2*t)= i;
180                         end
181                     end
182                 end
183             end
184
185             if ((area_petita~=0)&&(t==length(labels)))
186                 mini_counter= mini_counter + 1;
187                 u= [(positions_imatge(1)-positions_imatge(3)) (positions_imatge(2)-
    positions_imatge(4)) 0];
188                 v= [0 1 0];
189                 angle180=  atan2d(norm(cross(u,v)),dot(u,v));
190                 if (mini_counter >=4)
191                     angle_res180= angle180 - angle_update180;
192                 end
193                 angle_update180= angle180;
194             elseif (area_petita==0)
195                 mini_counter =0;
196                 angle_update180 =0;
197             end
198
199             if counter >2
200                 if (((positions_imatge_update(2*t -
    1)~=0)||(positions_imatge_update(2*t)~=0))&&(positions_imatge(2*t -
    1)==0)&&(positions_imatge(2*t)==0))
201                     if ((((area_update-
    (area_gran*0.2))<=area_gran)&&(area_gran<=area_update+(area_gran*0.2)))&&(disapear==0))
202                         disp('---------------->>Blob-Disappearence_Case_1<<-------------
    -')
203                         one_blob=1;
204                     else
```

```
205                         positions_imatge_update(3) = positions_imatge_update(1);
206                         positions_imatge_update(4) =positions_imatge_update(2);
207                         positions_imatge_update(1) = pos_x_image_petita_update;
208                         positions_imatge_update(2) = pos_y_image_petita_update;
209                         disp('--------------->>Blob-Disappearence_Case_2<<-------------
    -')
210                         one_blob=1;
211                         apear =1;
212                         cont_apear=0;
213                     end
214                 else
215                     cont_disapear= cont_disapear +1;
216                     if (cont_disapear>=4)
217                         disapear =0;
218                         cont_disapear=0;
219                     end
220                 end
221                 if (((positions_imatge(2*t -
    1)~=0)||(positions_imatge(2*t)~=0))&&(positions_imatge_update(2*t -
    1)==0)&&(positions_imatge_update(2*t)==0))
222                     if (((area_update-
    (area_update*0.2))<=area_gran)&&(area_gran<=area_update+(area_update*0.2))&&(apear==0))
223                         disp('--------------->>Blob-Appearence_Case_1<<--------------')
224                         one_blob=1;
225
226                     else
227                         positions_imatge_update(3) = positions_imatge_update(1);
228                         positions_imatge_update(4) =positions_imatge_update(2);
229                         positions_imatge_update(1) = pos_x_image_gran_update;
230                         positions_imatge_update(2) = pos_y_image_gran_update;
231                         disp('--------------->>Blob-Appearence_Case_2<<--------------')
232                         exep2 =1;
233                         one_blob=1;
234                         disapear= 1;
235                         cont_disapear=0;
236                     end
237                 else
238                     cont_apear= cont_apear +1;
239                     if (cont_apear>=4)
240                         apear =0;
241                         cont_apear=0;
242                     end
243                 end
244                 if (t==length(labels))
245                     area_update = area_gran;
246                 end
247                 if ((t==1)||(one_blob==1)) %The distance in pixels is found
248                     pos_x_image_res_gran  = positions_imatge(1) -
    positions_imatge_update(1);
249                     pos_y_image_res_gran = positions_imatge(2) -
    positions_imatge_update(2);
250                 end
251                 if (t==2)
252                     pos_x_image_res_petita  = positions_imatge(3) -
    positions_imatge_update(3);
253                     pos_y_image_res_petita = positions_imatge(4) -
    positions_imatge_update(4);
254                     exep1=0;
255                 end
256
257                 if (t==2) %it is checked the amount of blobs and finds out the actual
    distance
258                     if ((first_blob==1)&&(cont_first_blob==0))
259                         disp('First blob with x=0cm and y=0cm')
260                         cont_first_blob=1;
261                     elseif ((((positions_imatge(2*t -
    1)==0)&&(positions_imatge(2*t)==0))||one_blob==1)&&(first_blob)==1)
262                         if (exep2 ==1)
263                             pos_x_real = pos_x_image_res_petita * image2real_x;
264                             pos_y_real = pos_y_image_res_petita * image2real_y;
265                             exep2 =0;
266                             one_blob =0;
```

```
267                             angle_res180=0;
268                         else
269                             pos_x_real = pos_x_image_res_gran * image2real_x;
270                             pos_y_real = pos_y_image_res_gran * image2real_y;
271                             one_blob =0;
272                             angle_res180=0;
273                         end
274                     elseif ((positions_imatge(2*t -
    1)~=0)&&(positions_imatge(2*t)~=0)&&(first_blob==1))
275                         if (((-5 <= pos_x_image_res_gran)&&(pos_x_image_res_gran <=
    5))||((-5 <= pos_x_image_res_petita)&&(pos_x_image_res_petita <= 5)))
276                             pos_x_real = 0;
277                         else
278                             pos_x_real = ((pos_x_image_res_gran +
    pos_x_image_res_petita)/2) * image2real_x;
279                         end
280                         if (((-5 <= pos_y_image_res_gran)&&(pos_y_image_res_gran <=
    5))||((-5 <= pos_y_image_res_petita)&&(pos_y_image_res_petita <= 5)))
281                             pos_y_real = 0;
282                         else
283                            pos_y_real = ((pos_y_image_res_gran +
    pos_y_image_res_petita)/2) * image2real_y;
284                         end
285                     end
286                     if
    ((abs(pos_x_real)>=(size(ceiling,1)*image2real_x*0.25))||(abs(pos_y_real)>=(size(ceilin
    g,2)*image2real_y*0.25))||(accu_angle180_res>=80))
287                         disp('Unprecise distance')
288                     else
289                         if ((-1<=pos_x_real)&&(pos_x_real<=1))
290                             pos_x_real=0;
291                         end
292                         if ((-1<=pos_y_real)&&(pos_y_real<=1))
293                             pos_y_real=0;
294                         end
295                         fprintf('>YAY The robot moved %fdegrees\n',accu_angle180_res)
296                         if
    (((accu_angle180_res>=170)&&(pos_y_real_reverse>=8))||(angle180_exception==1))
297                             disp('---->CHANGE TO 180°<----');
298                             pos_y_real = pos_y_real*(-1);
299                             if (angle180_exception==0&&pos_y_real_reverse>=8)
300                                 angle180_exception=1;
301                             elseif (angle180_exception==1&&pos_y_real_reverse>=8)
302                                 angle180_exception=0;
303                            end
304                            pos_y_real_reverse =0;
305                         end
306
307                         if ((angle_exception_right == 1)||(angle_exception_left == 3))
308                             disp('---->CHANGE_RIGHT<----');
309                             pos_y_real_mom= pos_y_real;
310                             pos_y_real = pos_x_real;
311                             pos_x_real = pos_y_real_mom;
312                             angle_axis = 1;
313                             angle_cont = angle_cont + 1;
314                         elseif ((angle_exception_right == 2)||(angle_exception_left ==
    2))
315                             disp('---->CHANGE_180<----');
316                             pos_y_real = -pos_y_real;
317                             angle_axis = 0;
318                         elseif ((angle_exception_right == 3)||(angle_exception_left ==
    1))
319                             disp('---->CHANGE_LEFT<----');
320                             pos_y_real_mom = pos_y_real;
321                             pos_y_real = -pos_x_real;
322                             pos_x_real = -pos_y_real_mom;
323                             angle_axis = 1;
324                         end
325                         u= [(pos_x_real) (pos_y_real) 0];
326                         if (angle_axis==1)
327                             if(pos_x_real >=0)
328                                 v= [1 0 0];
```

```
329                             elseif (pos_y_real <0)
330                                 v= [-1 0 0];
331                             end
332                         elseif (angle_axis==0)
333                             if(pos_y_real >=0)
334                                 v= [0 1 0];
335                             elseif (pos_y_real <0)
336                                 v= [0 -1 0];
337                             end
338                         end
339                         angle=  atan2d(norm(cross(u,v)),dot(u,v));
340                         if (angle_axis == 1)
341                             if (pos_y_real < 0)
342                                 angle = -angle;
343                             end
344                         elseif (angle_axis == 0)
345                             if (pos_x_real < 0)
346                                 angle = -angle;
347                             end
348                         end
349                         fprintf('Computed %fdegrees\n',angle)
350                         fprintf('The robot moved x= %fcm and y= %fcm\n', pos_x_real,
    pos_y_real)
351                         accu_pos_x_real=accu_pos_x_real + pos_x_real;
352                         accu_pos_y_real=accu_pos_y_real + pos_y_real;
353
354                         accu_angle_res = accu_angle_res+angle;
355                         accu_angle180_res = accu_angle180_res+angle_res180;
356                         if(accu_angle180_res==accu_angle180_res_update)
357                             accu_angle180_res=0;
358                         else
359                             pos_y_real_reverse = pos_y_real_reverse +1;
360                         end
361                         if(accu_angle_res==accu_angle_res_update)
362                             accu_angle_res=0;
363                         end
364                         accu_angle_res_update = accu_angle_res;
365                         accu_angle180_res_update = accu_angle180_res;
366
367                        if ((accu_angle_res >= 91)&&(angle >= 4))
368                             if (angle_exception_left == 0)
369                                 if (angle_exception_right < 4)
370                                     angle_exception_right = angle_exception_right+1
371                                 end
372                             else
373                                 angle_exception_left = angle_exception_left-1
374                             end
375                         elseif ((accu_angle_res <= -91)&&(angle <= -4))
376                             if (angle_exception_right == 0)
377                                 if (angle_exception_left < 4)
378                                     angle_exception_left = angle_exception_left+1
379                                 end
380                             else
381                                 angle_exception_right = angle_exception_right-1
382                             end
383                         end
384
385                         % Plot the trajectory
386                         fprintf('Accumulated of x= %fcm and y= %fcm and %fdegree\n',
    accu_pos_x_real, accu_pos_y_real, accu_angle_res)
387                         ii = [ii accu_pos_x_real];
388                         jj = [jj accu_pos_y_real];
389                         hPlot = subplot(2, 2, 4);
390                         hold on
391                         plot(ii(:),jj(:));
392                         %xlim([-100 200])
393                         %ylim([-400 50])
394                         title('Robot Trajectory', 'FontSize', fontSize);
395                         grid on;
396                         xlabel('X(cm)')
397                         ylabel('y(cm)')
398                     end
```

```
399                 end
400             end
401             if ((2*t - 1)==1)
402                 pos_x_image_gran_update  = positions_imatge(2*t - 1);
403             elseif ((2*t - 1)==3)
404                 pos_x_image_petita_update  = positions_imatge(2*t - 1);
405             end
406             if ((2*t)==2)
407                 pos_y_image_gran_update = positions_imatge(2*t);
408             elseif ((2*t)==4)
409                 pos_y_image_petita_update = positions_imatge(2*t);
410             end
411         end
412
413     end
414 catch ME
415     % Some error happened if you get here.
416     disp('An unexpected error has occurred');
417 end
```